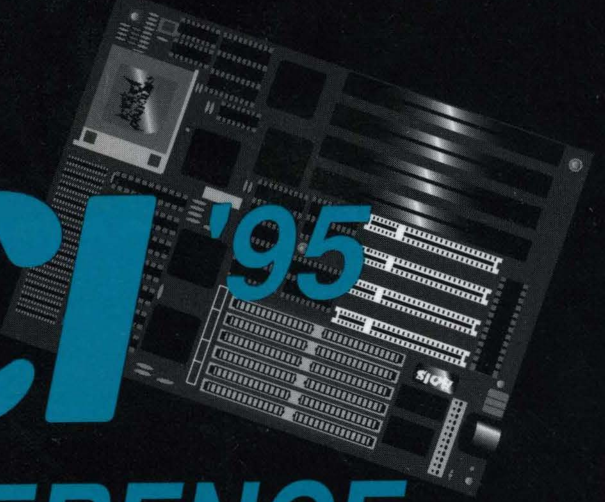
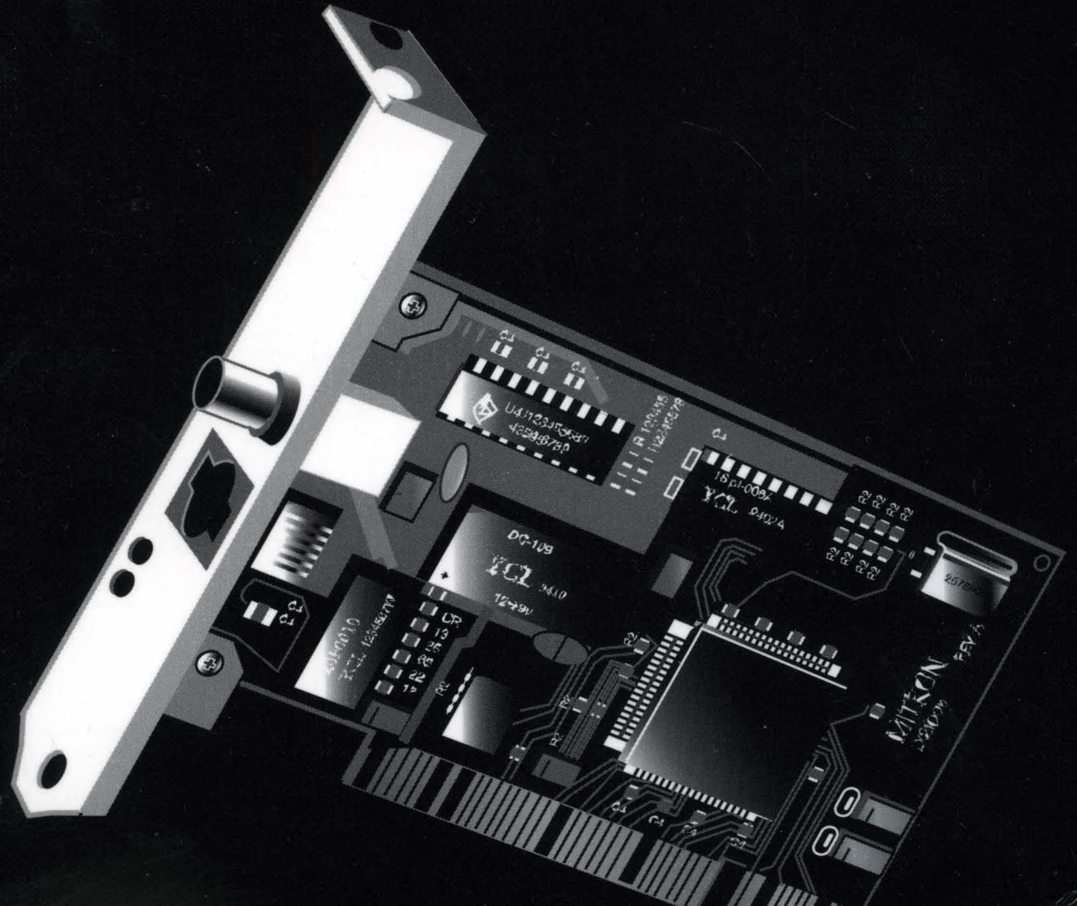
MARCH 27-31, 1995



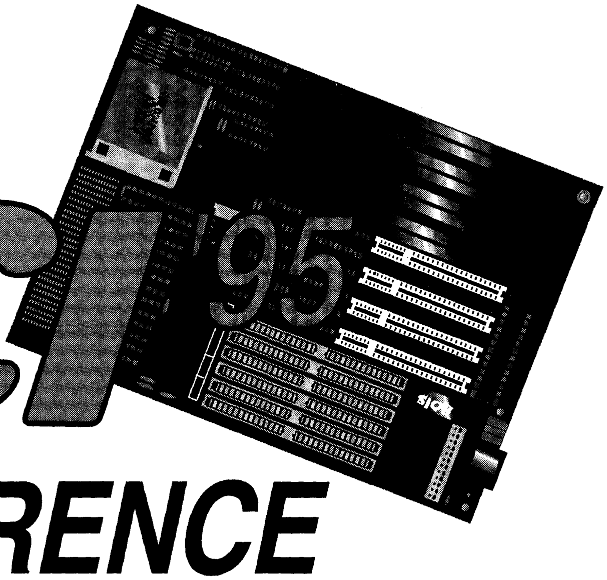
PCI '95

CONFERENCE PROCEEDINGS

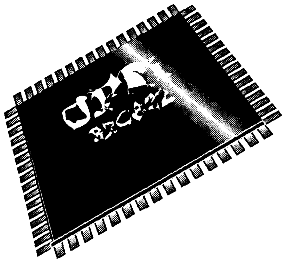


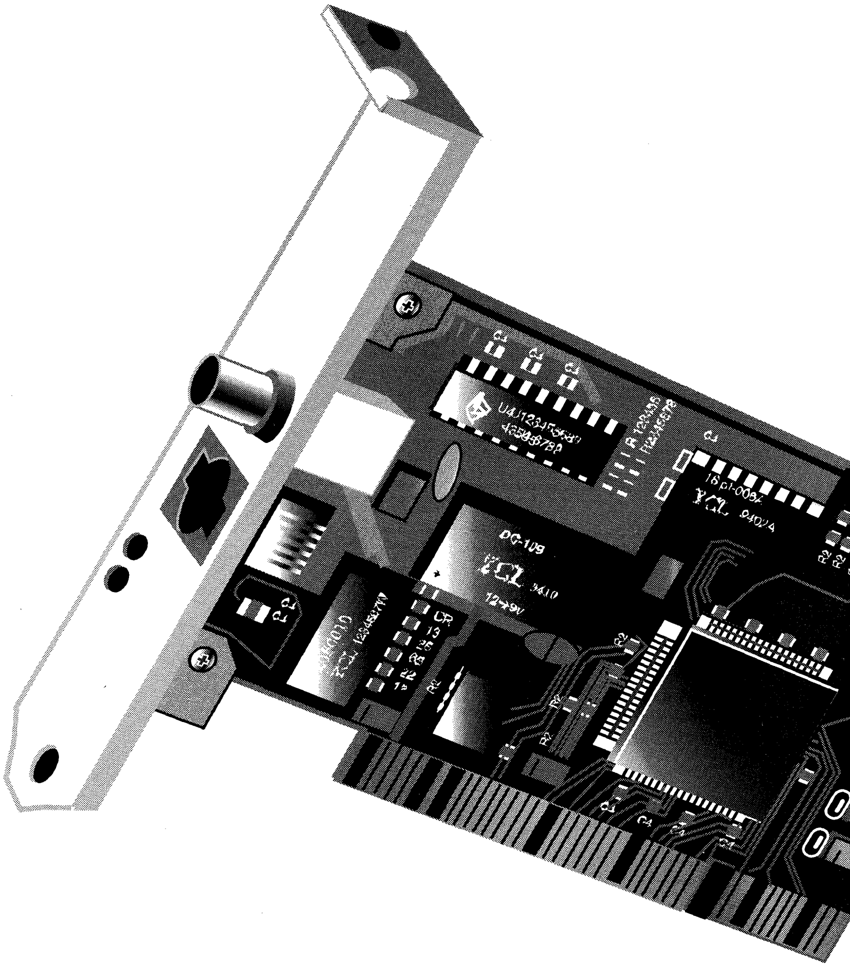
SANTA CLARA, CALIFORNIA

MARCH 27-31, 1995



PCI '95 CONFERENCE PROCEEDINGS





Annabooks
ISBN 0-929392-27-2

SANTA CLARA, CALIFORNIA

You are welcome to send us comments or questions concerning this or other Annabooks products, or to request a catalog of our other products and seminars.

Annabooks
11838 Bernardo Plaza Court
San Diego, CA 92128-2414

800-462-1042
619-673-0870
619-673-1432 FAX
73204.3405 @ compuserve.com

Proceedings of PCI'95 Conference
March 27-31, 1995 ● Santa Clara, California

TABLE OF CONTENTS

PCI'95 CONFERENCE
March 30-31, 1995 ● Santa Clara, California

SESSION 1A - WORKING WITH PENTIUM

SESSION 1B - GRAPHICS CARDS

SESSION 1C - CHIP MAKERS ROUNDTABLE

SESSION 1D - SYSTEM APPLICATIONS

SESSION 2A - CHIP SETS

SESSION 2B - NETWORK CARDS

SESSION 2C - COMPUTER MAKERS ROUNDTABLE

SESSION 2D - GRAPHICS CONTROLLERS

SESSION 3A - PCI MEZZANINE CONNECTOR DESIGN ISSUES

SESSION 3B - DISK CONTROLLERS

SESSION 3C - BOARD MAKERS ROUNDTABLE

SESSION 3D - SUPPORTING ISA LEGACY PERIPHERALS

SESSION 4A.1 - PCI ELECTRICAL DESIGN & BUS LAYOUT

SESSION 4A - SEMICUSTOM LOGIC IMPLEMENTATIONS

SESSION 4B - MOTHERBOARDS & BOARD COMPUTERS

SESSION 4C - LAN TECHNOLOGY

SESSION 4D - BUS TO BUS CONNECTIONS

SESSION 5A - SYSTEM LEVEL SOFTWARE ISSUES

SESSION 5B - MULTIMEDIA APPLICATIONS

SESSION 5C - POWER PC APPLICATIONS

SESSION 5D - SYSTEM DEVELOPMENT

SESSION 6A - THE FUTURE OF PCI

Late Submissions

Overview

Lance A. Leventhal, Ph.D.
Program Coordinator, PCI '95
Annabooks
11838 Bernardo Plaza Ct., Suite 102A
San Diego, CA 92128
Ph. (619) 673-0870 Fax (619) 673-1432

Since its introduction in 1992, PCI has rapidly become a standard in the world of high-speed personal computers. It has received the backing of virtually every major computer manufacturer, and a multitude of PCI chips, boards, and other equipment is now available. Major product areas include core-logic chip sets, bridge chips, interface chips, bus-to-bus connections, disk controllers, graphics cards, network cards, multimedia cards, board computers, motherboards (or system boards), and software. Test equipment manufacturers now provide extender cards, bus analyzers, logic analyzer add-ons, and other support. Models and other design tools are also available.

Industrial computer manufacturers have also moved toward PCI through the following efforts:

- The PMC (PCI Mezzanine Card) specification for a plug-in (mezzanine) connection between PCI and industrial buses such as VME, Multibus, and Futurebus. This work has the support of VITA, the VME Industrial Trade Association.

- The PCI Industrial Computer Manufacturers' Group (PICMG) specification for a backplane-based PCI system.

Other efforts include CardBus (derived from PCMCIA), Small Form-Factor PCI (SFFPCI), and Rugged PCI.

The advantages of PCI are now well-known. They include:

- 32-bit or 64-bit operation transparent to devices
- Allowance for either 5V or 3.3V boards
- Support for up to 132 MB/second data transfers
- Auto-configuration capabilities that eliminate the need for jumpers and DIP switches
- Compatibility with ISA, EISA, or Micro Channel systems and boards
- Operation at up to 66 MHz
- Processor-independence that allows for RISC-based as well as X86-based boards
- Detailed compliance specification and testing via PCI SIG and independent vendors
- Well-defined electrical and mechanical interfaces

The PCI '95 Conference Proceedings show the wide variety of products already available or planned. We can expect to see far more in the very near future.

ARCHITECTURAL AND DESIGN CONSIDERATIONS WHEN IMPLEMENTING HIGH PERFORMANCE MULTIMEDIA APPLICATIONS ON PCI

J. Scott Runner, Alak Deb, Maulin Bhatt, Dave Greenberg, Sean Ganjooi, Gina Ngo
Synopsys, Inc., DesignWare™ Components R&D
700 East Middlefield Road, Bldg. B
Mountain View, CA 94043-4033

ABSTRACT

Albeit PCI has been viewed as a key technology enabler in bringing multimedia to the desktop, server and peripheral market, there is a broad degree of latitude in the way in which multimedia may be implemented on PCI. A subspace of this design space allows for product differentiation and technology migration to avoid obsolescence;. On the other hand, another subspace does not provide for differentiation, but rather represents alternate design choices available to implement compliant systems. Some are more efficient than others, while some are better proven than others. The architectural and implementation tradeoffs when implementing various forms of video and graphics on PCI, their relative tradeoffs, and the implementation considerations associated with them will be explored in this paper. Pixel formats and details on addressing modes are not discussed in this paper. For more information on this topic, refer to (PCISIG, MM, 1994).

REQUIREMENTS FOR MULTIMEDIA

Before analyzing alternate architectures, we will determine the requirements for multimedia systems and add-in cards. This will be relative to customer, systems and card vendor and components companies to derive appropriate selection criteria, as follows:

Cost: Cost means more than the sheer sum of the components, board cost, testing, kitting/assembly and packaging costs, etc. It also includes the cost to support the product, to design it, and to re-design it (i.e., the cost to design a derivative product, or the next generation product). However, in the context of this paper, cost will refer to materials costs.

Upgradability: In the fast-paced, competitive market in which we live and work, the ability to quickly add functionality and take advantage of new H/W and S/W technologies is of paramount importance. Functionality should be scaleable in a straightforward manner between H/W and S/W. As an example, many video cards today provide for decompression and display of such data as real time video-conferencing or video clips off CD-ROM. But many of these cards do not provide access from the processor to the video frame buffer, and rely on video overlay via the VGA Feature Connector or VESA (Advanced) Feature Connector (V(A)FC). As processor bandwidth and video processing capability increases, the ability to leverage S/W for video functions will become potentially more valid and valuable.

Quality and Robustness: The promise of Plug and Play (PnP) and more standardized S/W-H/W interaction through standard APIs and H/W configurations is exciting. However, the demand that each device is placing on a system creates contentious situations between resources and challenging problems to standardize interoperability. Card designs should be efficient and fair while achieving compliance and must be robust and "fail-safe." No longer is it acceptable to just assert reset to solve a system dead-lock condition. Systems should avoid introducing errors, and should be non-destructively tolerant to their occurrence.

Performance: This is what has made PCI such an enabler. Tables 3a and 3b indicate the bandwidth requirements for various resources as compared with the available bandwidths of PCI.. While there is plenty of bandwidth, PCI will become more crowded as additional functionality and classes of devices migrate to this bus. It is important performance consider not just card performance, or local optimizations, but more importantly, that designers bear in mind optimization of the entire system to which they contribute.

ALTERNATIVE ARCHITECTURES

One of the advantages of PCI is that it supports a variety of bus and system architectures and configurations. This is also one of its challenges. Figures 1 through 4 depict alternate architectures for multimedia systems comprised at a minimum of a graphics controller, video source (compressed or raw), and audio source. Table 1 provides a comparison of these architectures. Having reviewed some of the key multimedia design requirements, attention will now turn to the design implementation tradeoffs for architectures described in Table 1.

A key point is that various architectures optimize different design parameters. None of the architectures is superior to any other in all cases, but rather each has specific advantages. While certain standards must be followed for the benefit of interoperability and PnP, there is a great deal of latitude available to the designers of multimedia systems that can provide product differentiation. Synopsys' DesignWare™ PCI MacroSet is a modular, parameterized synthesizable kit designed to facilitate fast implementation and exploration of this design space. With appropriate parameters, it can support all the configurations listed in Table 1 in a highly efficient and compliant manner.

Planar Bus Topology (Figure 1)

Some bus topologies require system changes, while others can be implemented entirely on the card. System modifications tend to take longer to implement and are typically expensive. Such architectures will appear first in workstations and embedded systems before we see them in high-end desk-side PCs, or even desk-tops. The planar bus will be around for quite a while, as the low cost system solution. However, card designers must keep in mind that as planar bus bandwidth becomes increasingly more consumed by higher bandwidth, more demanding functions, their cards must be efficient, and robust enough to gracefully degrade in the presence of traffic. Given that legacy busses will tend to bridge off this bus, and that CPU access to system resources is directly affected by this bus, cards placed on it must be not only robust and compliant, but also efficient as well. The 2.1 specification moves to address this by tightening up latency and wait-state tolerances and guidelines, but agents must still be tolerant of existing cards that may not be so efficient.

Cardtop/Feature Connector (Figure 2)

The VGA feature connector has become pervasive, and the VESA Advanced Feature Connector (VAFC) will be no exception. But note that the VAFC, as well as the Shared Frame Buffer Initiative (SFBI), are both complementary, not competitive with PCI. Such DAC attach circuits are simple to implement, support multiple video sources, provide high resolution, off-load video bandwidth from the graphics frame buffer, and keep their traffic off the system bus. However, they are expensive in that they introduce redundant system components (such as frame buffers) and do not allow CPU access to manipulate video.

Secondary Bus on a Card (Figure 3)

While this solution provides the same advantages as the secondary bus in the system, it leverages existing planar systems. It also avoids some redundancy in components, since the subsystem is entirely under the control of the card manufacturer. This applies to video addressing and pixel formats as well, and the optimization of the shared frame buffer. Due to the 10-load rule, this architecture does not allow for multiple bus connection from a single card. Rather, a PCI-PCI (or PCI-other) bridge must be employed to limit the load to 2. However, each function must incorporate its own PCI interface. Additional features may be added in future versions of cards either by higher integration of given functions on the card, or by adding functions onto the local bus.

This architecture is compatible with planar or hierarchical bus topologies and can be used today, or in tomorrow's systems. The drawback of this approach is that such cards will be relatively expensive, and many exhibit higher power consumption than the Integrated Multi-function card.

Since the card vendor controls all the functionality on the card, there may be a tendency to cut corners or restrict access to the card. Access from the CPU and other video sources is generally important, so be sure to provide for it via appropriate apertures (apertures are discussed in later sections).

Secondary Bus Between Cards (not pictured)

The hierarchical PCI bus is fast becoming an interesting architecture in implementing high performance multimedia systems to support multiple video streams, particularly for professional purposes. It is perhaps even more interesting to desk-side and server systems given the 10-load rule. If one envisions more than 4 PCI add-in cards, expect to see hierarchical bus architectures. While more expensive than the planar equivalent, this bus topology provides for isolation of multimedia components onto their own bus level, more controlled latency and bandwidth efficiency, and can off-load the planar when the graphics subsystem is moved to the secondary bus.

Integrated Multi-function Card (Figure 4)

Different from the secondary-bus on a card, the Integrated card has but one PCI Controller. This implementation may be PCI multi-function, in which each function logically has its own configuration space and slave at a minimum, or it may be implemented as a single function card in which arbitration, resource scheduling, datapath management and bus interface issues are all managed outside the PCI master/slave. In the former case, (a "loosely coupled" architecture), functions can operate independently of one-another, yet bus traffic, architecture, pixel formats, addressing and other issues related to efficient interoperability can be controlled. This means that functionality can be increased by adding components with their respective PCI interfaces. The latter case can be more powerful and cost efficient, requiring but one PCI controller interface.

While providing all the advantages of the previous architecture (Secondary Bus on card), this architecture provides for optimization of cost by avoiding redundant components, and system performance by off-loading bandwidth from other busses, while providing a tight coupling between multimedia resources.

DesignWare™-BASED MULTIMEDIA DEVICES

Synopsys has developed a PCI developer's kit that provides for fast, accurate and efficient implementation of PCI controllers for a broad variety of applications. This will be used as an example in the following implementation discussions. Note that block diagrams and specifications of these systems implemented with DesignWare™ are available from any of the authors.

Bus Functional Models (BFMs): These are BFMs of the PCI master and slave, as well as a passive monitor that observes and records bus activity and violations/exceptions and validates timing. Each model executes a series of commands test various compliance or user scenarios, and therefore be used for complete system tests.

Compliance Suites: These are a series of verification suites derived in collaboration with the PCISIG Protocol subcommittee to verify functional accuracy. Scenarios based on functionality supported by the unit under test are automatically generated for the appropriate configurations. Automatic post-processing of the test results creates a compliance report showing which SIG tests passed, which failed and/or are not applied.

DesignWare™ MacroSet: A set of seven Plug and Play, parameterized modules that can efficiently construct virtually any PCI implementation. In addition, DMA controllers, design examples, system-level test benches and Synopsys-optimized synthesis scripts are provided as a complete solution for the implementation of an on-chip PCI controller. Parameters allow a designer to quickly implement their required functionality, or explore the space of alternative implementations with rapidity, to quickly converge on optimal solutions.

Architectural Assumptions

It is assumed that the architecture will be an integrated multimedia ASIC that implements a PCI single function, single interface. We will assume that graphics, audio, video in and a H/W CODEC are all implemented on a single ASIC. These resources will access a single shared frame buffer, which will be accessible from other resources on PCI, including the processor. We will assume that the application-side domain operates at 50Mhz, and that we are dealing with 33Mhz PCI. Interaction between application components will be accomplished directly (i.e., the video CODEC dumping RGB16 data into the frame buffer will require no direct involvement from the PCI interface).

Point-to-point or "destination" addressing will be implemented with one H/W window for each video or graphics source. This is in accordance to the PCISIG Multimedia Design Guidelines. This provides support for a variable numbers of windows, support for H/W-S/W scaleable, and robustness and elegance in addressing pixel locations and dealing with buffer over-run and other errors. Apertures will be implemented to support YUV 4:2:2, RGB-15+a and RGB 24+a pixel formats, as well as big and little endian support. To allow access from external video sources, these apertures will be accessible from PCI bus resources as well. Resources on the PCI bus would access such spaces through apertures implemented in two ways with the DesignWare™ MacroSet. By defining a single address range and decoding the higher-order alias bits, or by assigning an address range and associated Base Address Register (BAR) for each aperture (PCI supports up to 6 such relocatable spaces) or a hybrid of the two. The MacroSet supports this by assigning a set of attributes to a range and simulating and synthesizing the implementation. In this case, a BAR is assigned to each pixel format, and a user-defined programmable bit associated with each BAR can be used to indicate "endian-ness". Note that this excludes concurrent support of multiple endian formats per pixel format.

An Address Recognition and Mapping (ARM) module decodes 'hits' into all address spaces, notifying the application of the space accessed, controlling the assertion of DEVSEL#, and performing address translation if it is specified. Space decodes can be used to enable apertures and/or resources, and to select among multiple datapaths. Typically one would want one FIFO or register for slave read, one for write, one for master read, one for write. Feeding the application from the slave on a slave write would amount to notifying the appropriate space that the data was available, and having the data read out before the next slave read transaction. An alternative implementation is to support one datapath FIFO per slave application resource. Therefore, one might have a command FIFO for the graphics engine, and one directly to the frame buffer. In this way, the FIFO can be read at anytime, either being notified when not empty, or at any programmable threshold and can service the FIFO when most appropriate.

Bandwidth Budgeting - Frame Buffer

Determining the required bandwidth of multimedia devices is critical due to their general real-time requirements and high bandwidth. Failure to do so may render a system inoperable, or result in poor quality graphics (dropped pixels, lines or frames) or audio, and may adversely affect other peripherals, such as disk controllers and LANs, which expect to have all the bandwidth they will need.

The frame buffer is a key resource. Not only must all local resources (display refresh, graphics, video, refresh) have sufficient bandwidth, but there must be sufficient residual bandwidth for external resources. Worst case demands will occur during an active display refresh line. Note that blanking can be leveraged to mask the effects of DRAM refresh (if required), as well as helping to resource-level non-real time accesses, such as those from the graphics engine after a period of peak activity. To do so requires that buffering from the interface to the frame buffer be sufficient to cover the available bandwidth during a scan line (or half a scan line on the average). This is because the PCI slave cannot guarantee that a disconnect-retry during an active line will resume during blanking.

As an example, if available FB bandwidth is 200MB/sec, and the card designed to support 1024 x 768 x 16 bpp resolution, then during a scan line at 72HZ refresh, 148MB/sec is consumed by display refresh. If a 320 x 240 True Color video image is being displayed at 30 FPS, then that consumes another 6.912MB/sec, leaving 45MB/sec to be shared between the graphics engine and the PCI bus. While the graphics engine is not 'real-time,' it can create a live-lock situation if the processor is waiting to write a graphics command while at the same time, buffering writes of real-time to the FB. Supporting a graphics command buffer that can buffer a number of commands that might be issued during a scan line is an ideal solution (alternately side band signaling of status between the graphics engine and the processor can work, but it's not as insensitive to bus topologies). Should command buffering or FB bandwidth become inadequate during peak activity, the graphics or available engine should gracefully degrade and not hang and certainly not affect real-time operation.

Latency and Datapath Buffering

Latency and Datapath Buffering Latency have always been controversial issues in the PCI community, partly because it has been affected by legacy busses so greatly, and partly due to the fact that latency and throughput have been somewhat at odds with each other, and high performance cards effectively want to optimize both. There are several components of latency, arbitration latency (which is typically 2 clocks for the highest priority device in the system); bus acquisition latency (the time from receiving a GNT# till the bus is IDLE and consequently available); and the target latency to respond to the transaction. Target latency for first data is now required to be 16 clocks; any more and the target must either perform a delayed transaction, or target-abort. (PCISIG, 2.1, 1994); Subsequent data transfers may see up to 8 clks to complete, but if the behavior of your target is known, the more accurate data would be factored into your calculations.

The controversial figure is that of bus acquisition latency which is very dependent on system configuration. The 2.0 specification recommends that 30µsec be the guideline for acquisition latency on the planar bus, while (PCISIG, 2.1, 1994) and (PCISIG, MM Guidelines, 1994) specify that bus levels off the planar bus typically would see less than 3µsec acquisition latency. This latency is affected by the value programmed into the master latency timer (MLT), as well as the typical and max burst duration of resources in the system. The number of masters and their priority also affect acquisition latency. For example, if there are 5 masters (max) on a bus level, each with their MLTs programmed to 32 clks, then if the target latency to first data was 16 clks and 8 clks between data transfers, then the acquisition latency assuming equal priority, could be as high as 32*(5-1)+8+16 = 4.56µsec of latency. The general rule is: tune your implementation to operate efficiently at 3µsec of latency, but insure that it can handle 30µsec of latency without catastrophic degradation. A few pixels dropped may be acceptable, while dropped frames or important control information are not.

Acquisition, generally the most variable and severe of the latencies, will require buffering in accordance with this equation:

$$S_{lat\_buf} = t_{LAT} * f_{arrival} * W_{data}$$

where t_{LAT} is the max latency time, $f_{arrival}$ is the frequency of the data arriving to be transmitted (for master writes), and W_{data} is the number of bytes per farrival. Note that t_{LAT} should include the maximum latency that the current master's last transaction may introduce (8 PCLKs), as well as the initial target latency of the new master (16 PCLKS under the 2.1 guidelines). As an example, a 50MHz application clock delivering WORDs will require 30µsec*50Mhz*2B/WORD = 3K bytes (min) of total buffering, while a 10baseT interface would require 30µsec*10MHz*1/8 = 38 bytes of buffering (min). This does not include master latency (from IDLE to FRAME# asserted) nor target transfer latency.

After the bus is acquired, the sourcing FIFO must be able buffer data at least to the following depth:

$$S_{xfer\_buf}=t_{tar\_xfer\_lat}*(f_{arrival}-f_{service})*(W_{data}/4 \text{ bytes/DWORD})$$

3K bytes of on-chip buffering is expensive, and often impractical, considering that less buffering is required after the bus is acquired. An effective solution is to partition the buffering into levels: one transfer FIFO to provide data sufficient for the longest, fastest burst expected, and a buffer store to cover acquisition and initial target latencies. Synopsys' datapath buffers are tightly coupled with the master to not only initiate transfers from populated FIFOs, but also to manage exceptions. These buffers are able to cycle data at 0 wait-states during a burst, for popular, modern ASIC technologies. Their depth should be computed based on the master's maximum burst size and the relative difference in the master-slave bandwidth (S_{xfer_buf} equation).

Sitting behind the master datapath buffer can be a store that buffers the amount of data necessary to cover acquisition latency and initial target latency and provide storage (if necessary) for other resources. This buffer could be implemented in SRAM, or DRAM. Transfer from this buffer to the master datapath buffers could be performed by a simple DMA controller such as the one provided with the DesignWare™ MacroSet. Note that while transfers are in progress, the buffer store can be back-filling the master's datapath FIFO, providing for a resource sharing of storage, reducing area.

If the master's data arrival bandwidth is sufficiently low in portion to the acquisition latency and target bandwidth and latencies, then a single buffer may be sufficient.

Master reads need not account for acquisition latency, but only the classic queuing theory case of the difference in arrival rate versus service rate. If a master read requests as much as X bytes of data, and the transaction is completed without interruption, in the worst case, no data was serviced by the application, then X bytes of master read FIFO buffering would be required. A similar analysis can be applied to the slave read and write buffering requirements. The DesignWare™ MacroSet supports this by providing for parameterized depth FIFOs and/or registers each way, for master and/or slave. Recent enhancements allow FIFOs to be constructed from Flip-Flops, or by leveraging ASIC vendor diffused or metal programmable SRAMs.

SUMMARY

Alternate PCI multimedia architectures have been reviewed, with emphasis placed on compliance, efficiency, and leverage the design implementation latitude that PCI provides. Where design decisions provide for differentiation, optimize them for one's own design criteria. However, be compliant and where design decisions are not differentiating, follow proven, conservative approaches, such as those specified in PCISIG Guidelines. Synopsys has viewed the various architectures presented herein as equally valid under differing design criteria, and have constructed a reusable, complete PCI design kit to support them.

Topology	Current System Effects	Card Cost	Bus Traffic Effects	Issues
MM on Planar	None Low cost	Low-moderate; (redundant commodity components)	• All bus traffic visible • MM traffic seen on system bus	• Must be very robust and gracefully degrade in presence of traffic • Common MM interaction protocol must be followed
Secondary bus between cards	Hierarchical bus support; Higher cost	Low; Existing card architectures need not change	• Confines multimedia bandwidth off of planar to secondary; seen by other secondary devices	• Common MM interaction protocol must be followed • Driver per card required
Secondary bus on Card	None Low Cost	High. Requires PCI-PCI bridge and bus on card; redundant logic	• Isolated from system bus traffic • Keeps MM traffic off of system bus if graphics on secondary	• Extensible - easy to add functions to new releases of cards • Can control MM interaction protocol • Driver per function on card
Cardtop or Feature connector	None Low Cost Usability & reliability?	Moderate (redundant components)	• Isolated from system bus traffic • Keeps MM traffic off of system bus	• Does not provide CPU access to video (not H/W-S/W scaleable) • Synchronization to display refresh rate for "live" video
Multi-function integrated card	None Low Cost	Moderate High features/$	• Isolated from system bus traffic • Keeps MM traffic off of system bus	• Can control MM interaction protocol • Local arbitration and merging/DMA important • One driver per card or function achievable

Table 1 - Comparison of Some Alternative Multimedia Bus Architectures

Resolution	Bits/plane	Frame Rate (FPS)	Bandwidth Budget (MB/sec)	Use
320 x 240	YUV 4:2:2 RGB 8	30	2.304	Video (SIF) Video - display
320 x 240	24 (True Color)	30	6.912	Video - display
352 x 480	YUV 4:2:2 RGB 8	30	5.069 15.208	NTSC - YUV NTSC - display
640 x 480	RGB 8 RGB 24	30	9.216 27.648	Video - display
640 x 480	RGB 8 RGB 24	72	22.118 66.355	Graphics - refresh
1024 x 768	RGB 8 RGB 24	72	56.623 196.869	Graphics - refresh
1280 x 1024	RGB 8 RGB 24	72	94.372 283.116	Graphics - refresh

Table 2a - Frame Buffer and Bus Bandwidth Requirements of Various Sources

Type of Frame Buffer Implementation	Cycle Time	Available Bandwidth (MB/sec)
32b-wide DRAM	40ns	100
	30ns	132
64b-wide DRAM	40ns	200
	30ns	264
64b-wide WRAM	20ns (fast page mode)	200 (parallel) + 114 (serial)
single channel RAMBUS	10ns	500

Table 2b - Bandwidth Availability of Various Frame Buffer Implementations

Type	Configuration	Peak Bandwidth Requirement (MB/sec)
320 x 240 Compressed Video Source	MPEG Compressed Full Window	.6
320 x 240 uncompressed Video Source	24b RGB True Color Full Window	6.912
SCSI Disk Controller	Fast SCSI-2 (8b/10mhz, synchronous)	10
	Fast-wide SCSI-2 (16b/10Mhz, synchronous)	20
	Serial SCSI	<= 100
Ethernet LAN	10baseT	1.25
	Fast Ethernet	12.5
ATM		19.375

Table 3a - PCI Bus Bandwidth Requirements of Various Sources

Width (bits)	Clock Domain (MHz)	Peak Bandwidth Available (MB/sec)	Mean Bandwidth Available[1] (MB/sec)
32	33	132	80
64	33	264	160
32	66	264	160
64	66	528	320

Table 3b - PCI Bus Bandwidth Availability

[1] Mean bandwidth depends on bus topology, number of cards and their bus traffic characteristics, arbitration behavior, effectiveness of bursting, bridge characteristics (ability to sustain burst without disconnect), etc. Also note that twice as wide doesn't mean twice as fast, since this depends on how effectively the data from the peripheral can be framed to QWORDs. Likewise, twice the clock rate does not mean twice as fast, since the peripherals must keep pace, given their own bandwidth behavior which may result in streams being broken up into a larger number of bursts. Therefore, this number is system specific, and meaningless in this context and is intended to simply show that there is plenty of bandwidth to be had. 32b, 33mhz systems have been constructed which demonstrate 80MB/sec sustained performance, but your mileage will vary. The key is to bear these issues in mind which architecting, so that the overall system is optimized.

The PCI Multimedia Design Guide and the PCI Local Bus Specification, Revision 2.1 (draft), provide important clarification's to the specification of latency and wait states, as follows:

Timing Parameter	2.0 Spec	2.1 Spec[1]	MM Recommendation
Addr to 1st Data - Writes	No Spec	16 PCLKs	1 PCLK
Addr to 1st Data - Reads	No Spec	16 PCLKs	No Spec
Wait states between data	8 PCLKs	8 PCLKs	0 PCLKs
Minimum Burst Length	No Spec	No Spec	8 PCLKs
Bus Access Latency	30µs	30/3µs	30/3µs[2]

Table 4 - Comparison of PCI Latency Specifications

[1] This references the draft specification, in which specifications may differ slightly.

[2] The Multimedia Guidelines stipulate that this specification applies to a secondary bus, and with a statistically high probability to the planar bus. This is especially true when bridges to legacy busses support specify there should be no more than 4 bus masters on the level of a MM device to insure the 3µsec latency guideline. Devices on the planar with ISA attach may see worst case latencies of up to 30µs.

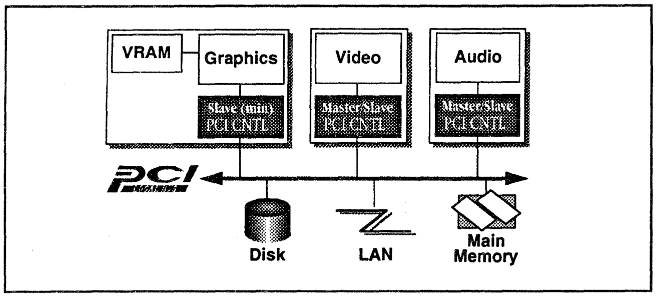


Figure 1 - Planar Bus Topology (separate cards)

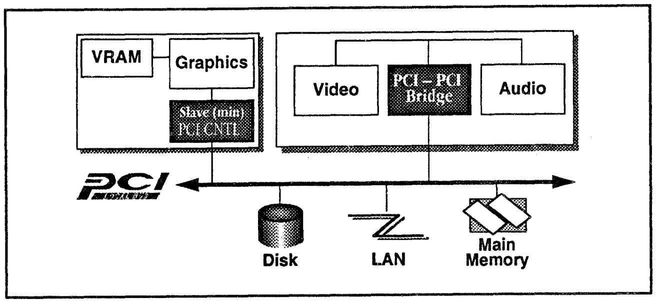


Figure 2 - Secondary Bus Between Cards

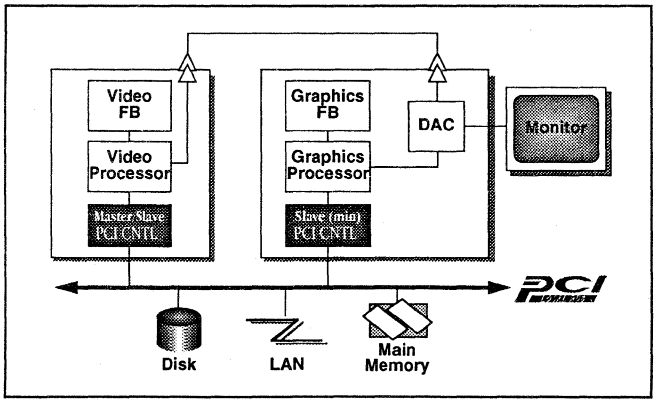


Figure 3 - Card Top/Feature Connector

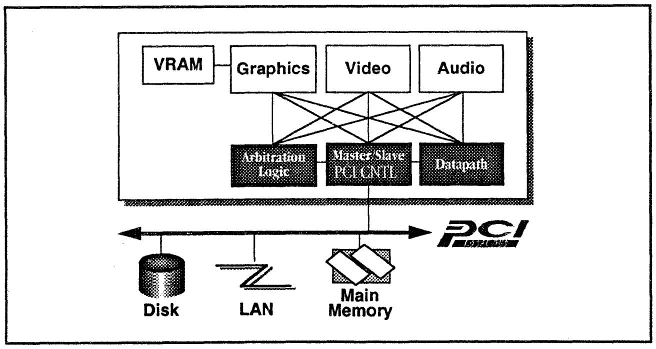


Figure 4 - Integrated Multi-Function Card

REFERENCES

Edward Solari, George Willse, PCI Hardware and Software (San Diego: Annabooks, 1994)

PCI Special Interest Group. PCI Developer's Conference Proceedings. April 20, 1994.

PCI Special Interest Group. PCI LocalBus Specification. Revision 2.0. April 30, 1993.

PCI Special Interest Group. PCI LocalBus Specification; Review Draft. Revision 2.1. October 21, 1994.

PCI Special Interest Group. PCI System Design Guide. Revision 1.0. September 8, 1993.

Synopsys®, Inc. DesignWare™ Components PCI MacroSet Databook. April 30, 1993.

RAMBUS TECHNOLOGY AND SYSTEM DESIGN OVERVIEW

Billy Garrett
Manager of Graphics Development
Rambus Inc.
2465 Latham Street
Mountain View, CA 94040
garrett@rambus.com

ABSTRACT

Rambus™ 500 MHz DRAM Technology removes the performance bottleneck faced by today's computer and graphics systems. Current computer systems use multiple banks of DRAMs and large SRAM caches to meet the performance demands of 486, Pentium, and RISC CPUs. To increase performance, DRAMs are arranged as multiple, interleaved banks on wide buses and are controlled by multiple, high pin-count ICs. Graphics systems use many VRAMs in wide, interleaved buses to meet the performance demands of Windows, video, and the new higher-resolution monitors. The high bandwidth, low pin-count Rambus solution enables highest performance at lowest system cost for high volume personal, portable and multimedia systems, and, as a result, is broadly supported in the industry.

This paper discusses the elements of Rambus Technology and their impact on graphics system design

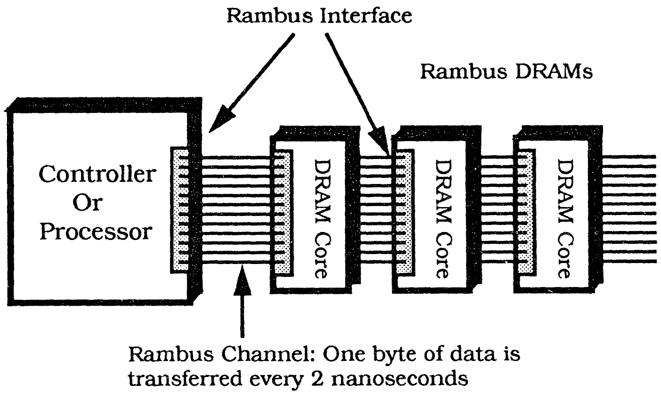
RAMBUS TECHNOLOGY OVERVIEW

The Rambus solution replaces the complex memory subsystem with a single, standard high-performance bus and Rambus DRAMs (RDRAM™). The Rambus solution has three elements: the Rambus Interface, the Rambus Channel, and the RDRAM.

The Rambus Interface is implemented on both the Channel controller and RDRAM devices. The controller directs the operations of the RDRAMs on the Channel through the use of a packet-oriented protocol. Rambus Channel controllers can be conventional microprocessors, peripheral chips, ASIC devices, memory controllers, or graphics engines.

The Rambus Interface on an RDRAM contains minimal logic and a few registers. This reduces die size overhead and maximizes cost effectiveness. The RDRAM is a CMOS DRAM incorporating unique architectural modifications and the Rambus Interface circuitry. The 16Mbit RDRAM is arranged as 2Mbitx8 or 2Mbitx9, while the 8Mbit RDRAM is organized as 1Mbitx8. Definition and use of the eight or nine data lines is left to the system designer.

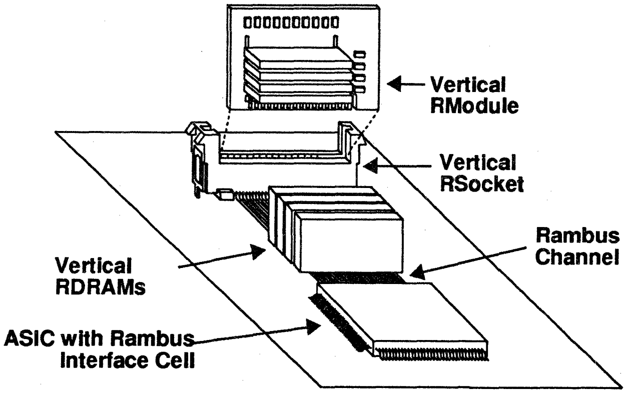
The Rambus Channel is revolutionary in that it is only eight or nine data bits wide, but is capable of transferring data at rates up to 500 MBytes per second from a single RDRAM. By contrast, today's fastest page-mode DRAMs transfer data at 33 to 50 MBytes per second.



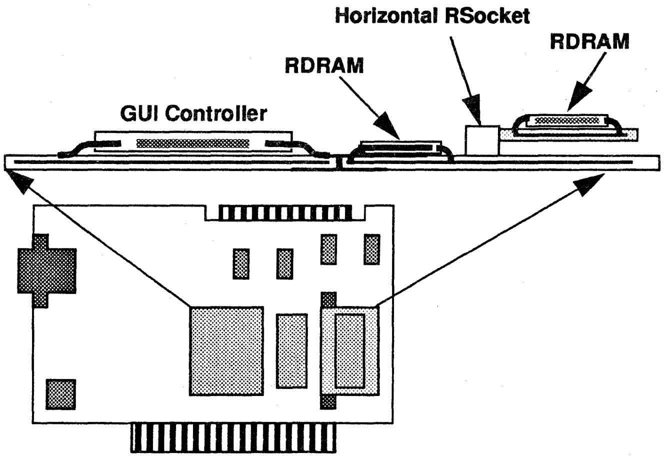
Rambus Technology Elements: Controller, Channel and RDRAMs

The Rambus Channel is also defined by a mechanical specification. The controllers and RDRAMs connect to the PC board with an interface that has only 15 active signals. The RDRAM package itself has 32 pins, including power and ground. The RDRAM is available in two plastic surface mount packages: an EIAJ standard vertical package (SVP) that allows dense packing for main memory applications, or a horizontal, low profile package (SHP) for add-in card applications.

The Rambus Interface transforms the data from the Channel's 2 nanosecond transfer rate to a 16 nanosecond cycle. It also converts the low-swing voltage levels of the Rambus Channel to the ordi-



A Rambus Motherboard Subsystem Example



A Rambus Add-in Card Example

nary CMOS logic levels used by the ASIC logic. The heart of this interface is a high-performance digital DLL (delay-locked-loop) circuit that provides the clocks for transmitting and receiving Rambus Channel data.

All critical system implementation issues have been resolved for the designer with the Rambus approach. Designers can implement a Rambus system following step-by-step documentation.

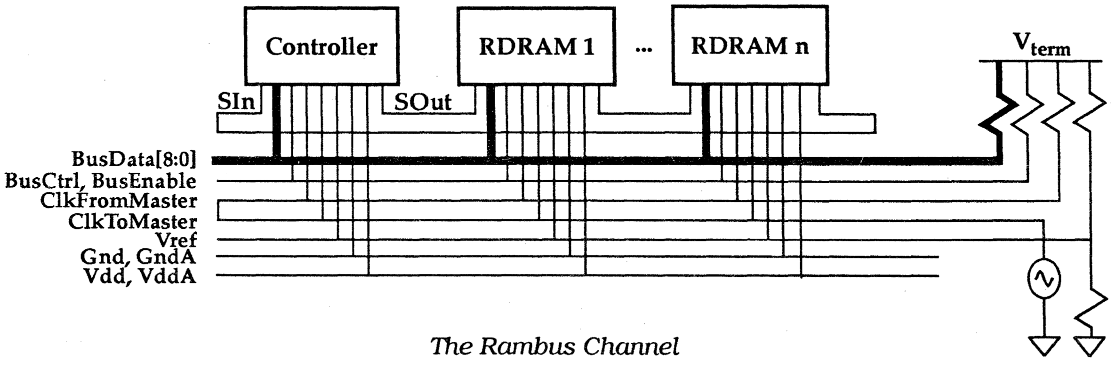
Example Subsystem Designs

The example subsystem figures above show typical physical implementations of a Rambus system including a controller that acts as a Channel master, a base set of RDRAM devices soldered directly on the Channel, and an RModule and RSocket used for memory expansion. At the heart of this system is the Rambus Channel itself. The Channel uses a small number of very high-speed signals to carry all address, data, and control information between Rambus devices.

The Channel is implemented using standard PC board layout and manufacturing techniques; it relies on controlled impedance terminated transmission lines to carry the high-speed, low-voltage-swing RSL signals. Clock signals are propagated in each direction on the Channel allowing data and clocks to always travel in parallel, virtually eliminating all clock to data skews.

Rambus Inc. has assured device independence by defining a high-level protocol that moves data in blocks and by using a large 36-bit address space. Designing new generations of hardware is simplified since the signals comprising the Channel will not change from generation to generation.

Each master and slave has its own Rambus Interface, which is currently available as an ASIC cell. This interface converts from the low-swing RSL levels used by the Channel to ordinary CMOS logic levels.



The Rambus Channel

Rambus Signaling Logic

The Rambus Channel achieves its high speed with dense packing, high-quality transmission lines, low voltage signalling, and precise clocking. A Rambus Channel contains controlled impedance, matched transmission lines:

- ❑ ClkToMaster
- ❑ ClkFromMaster
- ❑ BusData [8:0]
- ❑ BusEnable
- ❑ BusCtrl

These high-speed signals are terminated in their characteristic impedance. The Channel has a bus topology with the controller at one end, terminators at the other end, and the RDRAMs in between.

All high-speed signals on the Channel use low-voltage swings of about 800 mV. A logic "0" is equivalent to V_{term}, which is typically about 2.4V, V_{ref}, is about 2.0V and V_{OL} (logic "1") is about 1.6V. Within limits, all these voltage levels can be set by the system designer to control power consumption and noise margin. V_{ref} may be easily generated with a resistive divider.

V_{ref} sets the logic threshold for the high-speed RSL signals. This provides immunity from common mode noise on the Channel. All devices receive the low-swing signals with differential input circuits and use V_{ref} to set the logic threshold.

This differential sensing allows the Channel to use a low-voltage swing. Low-voltage-swing signals minimize dv/dt and, thus, di/dt to provide the following advantages:

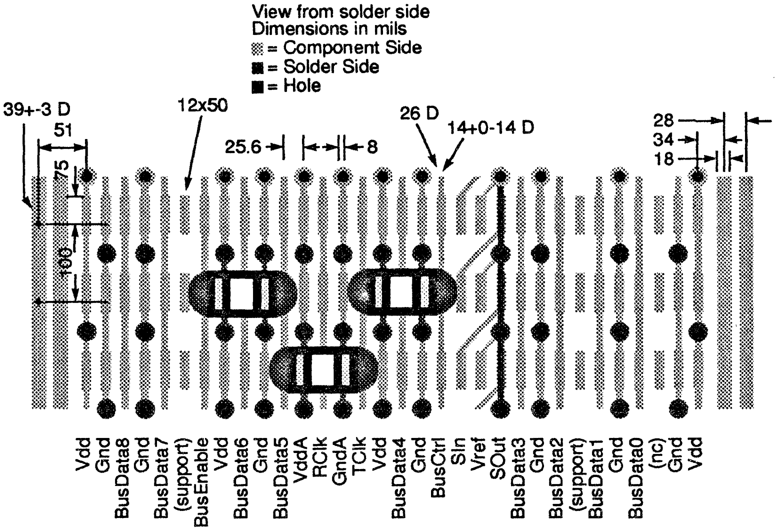
- ❑ Reduced ground bounce
- ❑ Reduced power consumption
- ❑ Reduced electromagnetic interference
- ❑ 5 volt and 3.3 volt device interoperability

Clocking

The Rambus Channel is synchronous, meaning that all data transfers are referenced to clock edges. At Rambus frequencies, special care has been taken to minimize clock to data skew for all RSL signals, as listed above.

The clock source typically is a separate oscillator. The clock routing begins at the slave end of the Channel and propagates to the master end as ClkToMaster, where it loops back as ClkFromMaster to the slave end and terminates.

This clocking topology allows clock and data to travel in parallel to minimize skew. A slave always sends data to the master synchronously with ClkToMaster, and the master always sends data to the slaves synchronously with ClkFromMaster. Because the transmission lines are matched, the clock and data signals remain synchronized as they travel to their destination.



Sample Channel Layout

Each Rambus Channel requires a (nominal) 250 MHz clock. The clock is driven at normal RSL levels. If a Rambus memory controller has multiple Rambus Channels, the clocks on each channel must be synchronized to arrive at the pins of the RAC with a minimum of skew.

Several vendors offer a pin-compatible clock driver designed to meet the needs of most Rambus systems. The 8-pin SOIC device connects directly to a low-frequency crystal and generates two copies of a Rambus compatible clock with less than 100 ps of skew between outputs. A series matching resister placed at the outputs allows it to drive bus impedances ranging from 20 to 50 .

Data Transfer

Direct data transfers occur only between the Channel controller and the RDRAMs. This allows signals to be terminated at only one end of the Channel. Data driven by the controller propagates past all RDRAMs with the desired voltage swing allowing all RDRAMs to correctly sense the data. The matched terminator prevents any reflections.

Data driven by an RDRAM moves in both directions at one-half the desired voltage swing. Data is effectively transferred on both edges of a 250MHz clock, resulting in a 500 Mbit-per-second-per-wire transfer rate. Each data transfer uses a 2 nanosecond interval, with 2 of these intervals per clock period.

Any one Rambus Channel is limited to approximately 10 cm. This length is determined by a 2 nanosecond propagation delay constraint for signals traveling from end to end on the Channel. As a result, a single Channel can accommodate up to 32 RDRAMs, 10 RSockets, or some combination of the two. Since each RModule can hold up to 32 RDRAMs, a fully configured system can have up to 320 RDRAMs, while a minimum system can have as few as one.

Component Packaging

Rambus has developed innovative packaging for the RDRAMs using proven materials and techniques. The packaging minimizes concerns such as on- and off-chip impedance that could arise when transferring data at 500 MByte per second speeds. Two styles of packages are available: the vertical "SVP" package allows very dense packaging for motherboard-based memory subsystems; the horizontal "SHP" package is optimal for low-profile add-in card subsystems. All Rambus Interface pins are located on one edge of the plastic package, thus aligning die pads, package leads, and printed circuit board traces, while minimizing the length of leads and bond wires. As a result, PC board electrical and interconnect issues such as inductance, capacitance, traces, and connectors are managed efficiently, enabling the Rambus solution to achieve 500MHz data transfer rates.

Rambus memory systems are expandable. Because the Rambus Channel and DRAMs were developed with expandability in mind, there are now simple, low cost Rambus Sockets (RSocket™) available from Augat Inc. and Molex Corporation that are supported by Rambus Modules (RModules™). RModules can support one to 32 RDRAMs. The motherboard subsystem and add-in card example figures shown earlier illustrate the use of vertical and horizontal RSockets and RModules.

System Packaging

Printed circuit boards carrying Rambus Channel signals are designed using standard FR-4 construction. Dielectric thickness is 5 mils (surface trace to ground layer) with 8 mil copper traces, resulting in a nominal 55Ω trace impedance. This impedance needs only be controlled to within a ±20% tolerance during bare-board manufacturing.

Separate power and ground planes are required for noise immunity. Two or more signal layers may be used. Design rules call for 8 ±1 mil wide signal traces on 0.65 mm (about 25 mil) centers. This spacing matches that of pins emanating from packages incorporating a Rambus Channel. The "Sample Channel Layout" is an example of a board design that includes a Rambus memory subsystem. Rambus Inc. supplies detailed information for use by PC board layout personnel; this "cookbook" streamlines the layout process. In addition, due to its smaller number of signals, a Rambus Channel design generally uses fewer PC board layers than traditional designs.

GRAPHICS SYSTEM DESIGN ISSUES

Rambus DRAMs provide the highest performance and lowest overall system cost of today's PC graphics. As shown in the table, RDRAMs provide sufficient drawing bandwidth for all popular and professional-level screen resolutions. Rambus-based graphics cards are cost-effective from both component and system-oriented viewpoints.

Graphics frame buffers for popular resolutions of 1024x768x8bpp require hundreds of megabytes-per-second of bandwidth for refreshing the display and accelerating graphics operations such as line drawing, shading, and text. Rambus DRAMs provide highest performance and lowest system cost for PC graphics. 500MHz Rambus DRAMs are able to satisfy the graphics display bandwidth requirements for a single component. The 1 or 2 MByte graphics subsystem is reduced to the GUI accelerator and a single DRAM component. Rambus DRAMs are based on 16Mbit DRAM technology which is lower cost per bit than 4Mbit technology starting in 1995. Rambus Technology provides a complete system solution; there is no need for additional registers, buffers, or glue logic.

Table 1: Display Resolution versus Frame Buffer Requirements

	Resolution	Drawing Bandwidth *	Frame Buffer Storage
Single Rambus Channel	800x600x8 800x600x16 1024x768x8	250-300 MB/sec	1 of 8Mb RDRAM
Single Rambus Channel	800x600x24 1024x768x16 1280x1024x8	200-250 MB/sec	1 of 16Mb RDRAM
Dual Rambus Channels	1024x768x24 1280x1024x16 1280x1024x24	600-700 MB/sec	2 of 16Mb RDRAMs

* Bandwidth Available After Refresh

A Rambus DRAM is the only 16Mbit-based DRAM to provide sufficient bandwidth for professional-level graphics products. Alternative DRAM approaches to achieve this level of bandwidth require the use of multiple 4Mbit-based DRAMs organized in wide, 64-bit data buses. As mentioned above, 16Mbit DRAM technology has become lower cost-per-bit than 4Mbit technology. A single 16-Mbit RDRAM takes up far less board area than the alternative solution: four 256K x 16 SDRAMs.

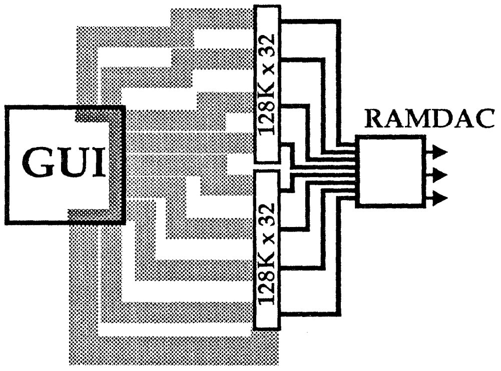
A Rambus-based graphics subsystem allows cost savings in the graphics controller component. The Rambus interface requires only 31 pins on the graphics controller. This is 80 to 100 fewer pins than the number required to implement a controller supporting the alternative 64-bit wide data path to the frame buffer. This lower pin count allows GUI controller designers to realize die-area and packaging-cost savings, while enabling them to take advantage of additional available on-chip real estate and pins to add multimedia features. A savings of 80 pins on a controller package can result in halving the die area of the controller, and greatly reducing die costs.

COMPARATIVE ANALYSIS

Compared to the Rambus memory solution, alternative approaches using multiple DRAMs or VRAMs in wide 32- or 64-bit parallel buses to provide the bandwidth required by 1024 x 768 x 24bpp displays face a number of system issues. In particular, the wide bus interface used with 32- and 64-bit parallel buses requires more DRAM components and controller pins than the Rambus approach. Designers looking to integrate multimedia features into their conventional designs have two choices: to move to a larger and more expensive controller package with more than 240 pins, or to develop a solution that requires multiple chips.

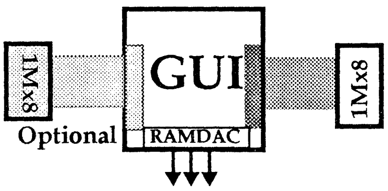
In the case of a conventional single-ported DRAM architecture, a 105-pin memory interface may lead to a larger controller package and more

Dual-Ported DRAM Approach



- ❑ 110 pin interface to memory
- ❑ 200 MBytes per second max drawing bandwidth
- ❑ Complex PCB layout
- ❑ Partitioned display bandwidth

RDRAM Approach



- ❑ 1 MB to 8 MB frame buffer
- ❑ Internal RAMDAC
- ❑ 450 - 900 MBytes per second bandwidth
- ❑ Second Channel can be added

complex board layout—all of which yields a maximum bandwidth range of from 200 to 400 MBytes per second. In contrast, the Rambus solution consists of a single-chip, two-megabyte frame buffer, smaller packaging, more room for additional features, and considerably higher bandwidth ranging from 400 to 450 MB/second.

Graphics designers have traditionally moved to dual-ported DRAMs, such as VRAMs, in order to gain higher performance than single-ported DRAM approaches. A dual-ported VRAM design, with its 110-pin memory interface further complicates the board layout considerations and may requires the need for larger controller package. The current trend for today's PC GUI controllers is to include the RAMDAC on-chip: a design that require even more pins on the controller if the VRAM's serial ports are connected back to the controller to the internal RAMDAC. The Rambus approach provides the ability to add a second Channel, increasing effective bandwidth of up to 900 MBytes per second, enough to support even the most advanced PC graphics applications.

The high integration and low-cost benefits of Rambus Technology are passed on to PC graphics subsystem board designers. A Rambus-based frame buffer requires just a few square inches of board space—about one tenth the space required by conventional graphics board designs. The Rambus minimal board area also helps to relieve the crowded Pentium motherboard: the Pentium's SRAM cache and DRAM backing store can be replaced with a Rambus memory subsystem with no loss in performance.

BIOGRAPHY

Billy Garrett is the Manager of Graphics Development at Rambus Inc. Rambus Inc. has developed the 500MHz Rambus Technology for high volume personal, portable, and multimedia systems. Mr. Garrett holds both a BSEE ('82) and MBA ('88) from the University of South Carolina. He has worked on a variety of design projects including: PC graphics boards, semi-custom ASICs for graphics and main-memory applications, PC systems, UNIX servers, and X-Terminals. Mr. Garrett holds several patents in these areas.

BIBLIOGRAPHY

"Applying Rambus Technology to Graphics," Rambus, Inc. Publication, 1994.

"Rambus Architecture Overview," Rambus, Inc. Publication, 1994.

"Rambus Product Catalog," Rambus, Inc. Publication, 1994.

TRADEMARKS

Rambus, RDRAM, RSocket, RModule are trademarks of Rambus Inc. All other brand and product names used may be trademarks of their respective companies.

OEM System Tuning Guide for Pentium™ Processor-Based Computers

Fred Pollack
Director of Measurement, Architecture, and Planning
Intel Corp.
Microprocessor Products Group
5200 NE Elam Young Pkwy.
MailStop JF1-91
Hillsboro, OR 97124
(503) 696-4953
(408) 765-4423/5947 (fax)
pollack@ichips,intel.com

This presentation identifies software and hardware enhancements that improve overall system performance of a 90 MHz Pentium processor-based desktop system. Each system component was changed systematically to determine the effects on overall performance. Specifically, various I/O bus implementations, graphics resolutions, device drivers, memory hierarchy schemes, and disk caching methods were evaluated. This information is meant to benefit and guide OEMs in selecting peripherals and system design strategies to achieve optimum performance. What we find is the need to pay special attention to all aspects of system design, including software device drivers, in order to obtain optimum Pentium processor performance as measured by the SYSmark93 for Windows benchmark.

PROGRAMMABLE LOGIC IN PCI-BASED SYSTEMS: AN OVERVIEW

David J. Ridgeway

Xilinx Inc.
2100 Logic Drive
San Jose, California, 95124

ABSTRACT

The Peripheral Component Interconnect (PCI) bus eliminates the I/O bottlenecks of traditional system buses by providing a high performance datapath for system CPUs and peripherals to communicate. This is essential for computation-intensive applications such as sophisticated graphics, local-area networking and real-time video which require large amounts of data processing and high speed system throughput.

This paper will focus primarily on hardware systems design with specific emphasis placed on PCI bus interface implementation and verification issues. We will first look at the background, features and functionality of the PCI bus. This will be followed by a design example, showing how system requirements drove the selection of specific technologies in order to achieve particular performance and marketing goals.

WHY PCI LOCAL BUS

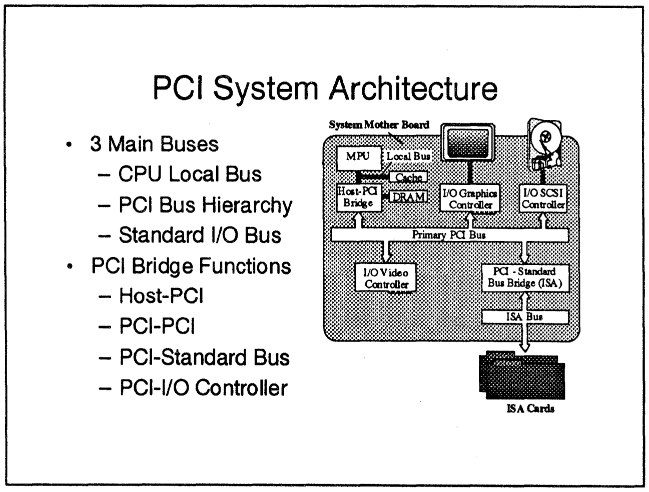
A personal computer system bus performs the task of moving data between the CPU and the peripherals such as disk drives, monitors, and printers. The most successful of these has been the 16-bit Industry Standard Architecture (ISA) bus established by IBM and its 32-bit successor, the Extended Industry Standard Architecture (EISA). However, the capabilities of new high-speed CPUs such as Intel's Pentium processor and the high data throughput requirements of applications such as graphics and video processing have quickly exceeded the data transfer capabilities of these standard system buses. As a result, many systems today are adopting one of the newer "local bus" standards as a means for improving overall performance.

By circumventing the I/O expansion slots, local bus peripherals tap directly into the path between processor and motherboard. The PCI Local Bus accesses the processor's local bus through a bridge. Because it is independent of the system bus, it is possible to use the local bus to achieve the performance required for critical functions such as video, and still maintain compatibility with existing peripheral hardware through the system I/O bus.

Although PCI delivers performance similar to a direct processor connection, it is in fact physically removed from the processor bus by a PCI Bridge. This bridge places a managing layer between the CPU and peripherals - creating a uniform interface. This also provides support for bus mastering, enabling intelligent devices to directly access main memory. PCI also includes an optional burst mode that provides accelerated throughput of data across the bus.

PCI defines four bridge types:

- *Host to PCI Bridge* - Connects the host CPU, memory and cache subsystem to the PCI bus.
- *PCI to Standard Bus Bridge* - Connects PCI to standard I/O bus such as ISA.
- *PCI-to-PCI Bridge* - Provides a connection path between two independent PCI buses, allowing a hierarchy of multiple PCI buses.
- *I/O Controller* - Translates between the PCI bus and the I/O protocols.



PCI System Architecture

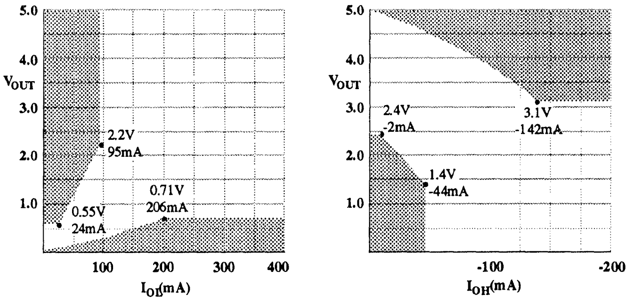
- 3 Main Buses
  - CPU Local Bus
  - PCI Bus Hierarchy
  - Standard I/O Bus
- PCI Bridge Functions
  - Host-PCI
  - PCI-PCI
  - PCI-Standard Bus
  - PCI-I/O Controller

PCI BUS TECHNOLOGY

The PCI bus is unterminated, operating on the principal of reflected wave signaling. The output impedance of a device driving the bus is roughly matched to the characteristic impedance of the bus (a transmission line). The incident wave from the output driver travels down the bus, reflects off the unterminated end and travels back to the receiver where the voltage doubles to meet the required input threshold. PCI defines a worst case signal-propagation delay (tPROP) of 10 nsec, one-third of the clock period for 33-MHz operation.

To drive the transmission line, IC output drivers must source and sink a minimum amount of current to ensure a large enough step on the line, given the characteristic impedance.

Loading and I/O Drive

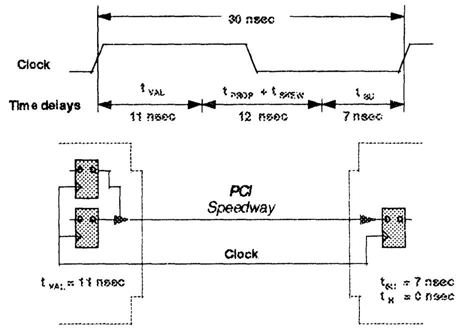


The PCI V/I drive curves show that the minimum impedance an output must be able to drive is 31.8 Ω (1.4 V/44 mA) in the logic high state. All output drivers for PCI components must satisfy these minimum / maximum drive characteristics to ensure sufficient signal drive during AC switching.

PCI employs a 32-bit multiplexed address and data path, which provides a peak bandwidth of 132 Mbytes/sec at 33 MHz. The basic data transfer mechanism on the PCI bus is a burst, comprised of an address phase followed by one or more data phases.

The PCI bus 30 nsec clock period allocates 11 nsec for clock to out delay (tVAL), 10 nsec for the bus propagation delay (tPROP), 2 nsec for clock skew (tSKEW), and 7 nsec for input set-up time (tSU).

PCI Bus Timing Requirements



The PCI bus has a number of signals that must be driven by more than one bus agent during a data transaction. To avoid bus contention, PCI requires bus turn-around cycles, or high-impedance states, at each point where a driving agent releases control of the PCI bus. This must occur one clock cycle before another agent can begin driving the bus to avoid contention.

An additional complexity in this scheme, is that the turnaround cycles occur in different bus cycles for different classes of signals, resulting in a need for multiple, independent output buffer enables. Target bus agents require five independent output enable controls while combined Initiator/Target bus agents require seven.

PCI protocol requirements are especially difficult to meet in chip architectures where the output enable signal must be driven from an internal flip-flop. Determining device compliance to these requirements is difficult. It requires careful analysis of device AC timing parameters, along with clock distribution, and internal propagation delays through the interconnect and logic structures.

This completes the overview of the features and functionality of PCI. We will now take a look at a design example focusing on the choice of technologies needed to meet this design's requirements.

VIDEO DIGITIZER DESIGN EXAMPLE

MuTech Inc. specializes in high resolution PC-based image processing solutions for OEMs in machine vision, laboratory image analysis, and color publishing. The firm has developed a series of single-monitor and dual monitor Image/VGA frame grabber boards that provide high-quality, low noise, color and gray-scale digital video images used in finger print identification, gel electrophoresis, parts inspection, and biotech cell classification.

In order to develop the next generation of image peripherals, MuTech was faced with the challenge of developing a high performance, low cost system that could capture and transfer high quality images quickly. MuTech selected the PCI Local Bus based on its high throughput capabilities. The PCI bus is so fast that images can be captured and transferred to system memory in real-time without the expense of large on-board memory. Using PCI, MuTech targeted continuous transfer rates of more than 20 Mbytes/sec from the video digitizer to system main memory and more than 30 Mbytes/sec to the VGA Display. This was over 10 times the performance level of an ISA bus based video digitizer.

SELECTING A SILICON SOLUTION

Attaching a peripheral device to the PCI bus requires an I/O controller chip that implements the PCI protocols on the PCI side and connects to a back-end bus on the reverse side. Determining the best product for the interface requires careful understanding of PCI design requirements. This includes protocol timing requirements, I/O drive characteristics, and the implementation of PCI autoconfiguration registers for Plug and Play operation.

Standards such as IDE disk drives, Ethernet LANs, and SCSI devices define the controllers for these applications and many vendors offer these controllers with PCI interfaces. Unfortunately, few off-the-shelf products have been designed to support the need for a general purpose I/O controller. To attach non-standard devices to the PCI bus, designers have limited choices and in many cases will have to design their own general-purpose PCI controller.

An important decision in a custom PCI interface design is whether you will implement the design using chipsets, gate arrays, or programmable logic. The chart below compares some trade-offs for each. Programmable logic technology has advanced such that performance is competitive with ASIC designs for many applications.

There were several factors that were key for MuTech in deciding which technology to use. Besides the need for 100% PCI-compliance, MuTech required integration of specific features that no existing chipset supported. It was critical to be able to prototype and experiment with multiple iterations of the design without worrying about costly NRE charges and prototype cycle times. Time-to-market was important as well as the ability to customize the final product for specific host systems. These factors led MuTech to select a Field Programable Gate Array (FPGA) as their solution.

One of the most important feature requirements of the design is that it be easy to install and configure. As a true Plug and Play PCI add-in, the board memory and registers are all automatically assigned addresses by the system BIOS at boot-up thus resolving the memory, I/O base, and IRQ conflicts that have plagued traditional ISA bus cards.

	Chipset	Gate Array	FPGA
PCI Compliance	Yes	Yes	Yes
Customization	None	Full	Full
Design Flexibility	None	Costly	Fully Reconfigurable
NRE	None	High	None
Prototypes	N/A	Lead time/Costly	Fast/Free
Time-to-Market	Fast	Medium	Fast
Availability	Immediate	Lead time/Risk	Immediate
Testing	Factory	User Supplied	100% Factory
Unit Cost	Low	Low	Masked Migration

DESIGNING FPGA-BASED PCI INTERFACES

Using a high-density FPGA, MuTech engineers were able to integrate the video capture control, VRAM memory control, board control registers and the complete PCI Interface into a single device. This high level of integration yields several benefits including high-performance, small package size, and low power consumption.

The Xilinx XC3100A family provides all the necessary features and logic capacity required to integrate the interface requirements onto a full sized PCI bus circuit board. The XC3100A architecture contains three major circuit blocks which were used to develop this design:

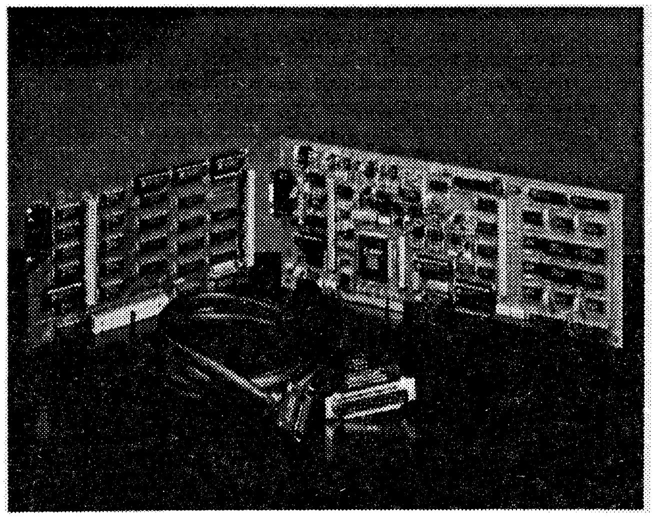
- *Input / Output Block* (IOB). Provide registered and buffered input as well as 3-state registered or buffered outputs.
- *Configuration Logic Block* (CLB). 5-input logic functions with registered or combinatorial outputs.
- *Channeled Routing*. Signal routing is accomplished using internal horizontal and vertical routing resources.

Implementing the design to run at the full burst rate of 33 MHz required a high degree of pipelining to optimize the routing and logic delays of the bus interface and memory control blocks. Registered input and output buffers were used on both the PCI and VRAM interface ports of the design. The resulting pipelined data flow increased the initial PCI access latency but produced a design that performs burst transfers at 33 MHz. For write burst, PCI address/data signals are latched by registered input buffers on one clock edge, and appear on the VRAM data port one clock later. A similar data flow occurs in the reverse direction on read cycles.

In addition to pipelining, partitioning the logic to utilize the 5-input CLB structure eliminated multiple CLB delays and maximized logic utilization. To meet the high I/O demands of a PCI controller, the 160-pin XC3195A package provides 138 I/Os. The XC3100A-2 family I/Os have been characterized to guarantee PCI compatibility. The devices have CMOS outputs with a guaranteed static output current of 8 mA. PCI dynamic drive requirements are not covered by the original data sheet, however, Xilinx has characterized devices and verified their compliance in all Plastic Quad-Flat Pack (PQFP) packages.

MV-1000 VIDEO DIGITIZER

This is a picture of the completed design. On the right is the video digitizer board (MV-1000) containing the PCI interface, analog camera interface, video capture, video conditioning and VRAM buffers. On the left is an optional digital camera interface board (MV-1100) which is EIA 422 compliant and allows the board to support up to four concurrent 8-bit digital inputs.



The entire system fits into a single full-sized PCI card slot. Connectors at the back of the MV-1100 enable the MV-1000 to support most of the popular higher resolution line scan, area scan, and variable scan cameras sold by vendors such as Kodak, Dalsa, EG&G, Ektron, Pulnix, and Roche.

SUMMARY

MuTech set out to design a high performance video digitizing system with a broad potential market. The system bandwidth requirements made the PCI bus a natural selection. In addition, because PCI is processor independent, designing a PCI based video digitizer would give them the potential to sell one board into PC, PowerPC and workstation platforms.

The bus interface and digitizer block required high performance with 100% PCI compliance. It required a high-level of integration and the ability to experiment with the design by testing multiple iterations during the design phase. Finally, time-to-market was critical. Together, these requirements drove the selection of an XC3100A FPGA as the technology of choice for implementing the critical functions of the system.

Designing a Flexible PCI Bus Interface with Programmable Logic

Bradly K. Fawcett
Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124

ABSTRACT

PCI-compliant high-density programmable logic devices can be used to create flexible PCI bus interfaces while avoiding the costs and risks of custom IC development. However, careful design is required to meet the performance and signaling requirements of the PCI specification. This paper overviews these issues, focusing on the attributes needed in the programmable logic device to facilitate interface design, and suggesting appropriate design techniques and methodologies. Examples of PCI-compliant devices from the Xilinx product line also are examined.

INTRODUCTION

The Peripheral Component Interconnect (PCI) bus definition has the admirable goal of providing high throughput on a well-defined, lightly-loaded bus while being compatible with today's IC manufacturing processes. In order to insure interoperability in all PCI-based systems, components that interface to the PCI bus must strictly adhere to the PCI specification's performance and I/O drive requirements. Thus, equipment manufacturers planning to design PCI-based systems and add-in boards face the challenge of locating and selecting PCI-compliant devices.

With the recent emergence of PCI-compliant Field Programmable Gate Arrays (FPGAs) and EPROM-based Programmable Logic Devices (EPLDs), designers of PCI bus interfaces can now reap the flexibility and time-to-market benefits of high-density user-programmable devices, while avoiding the costs and risks of custom and semi-custom IC development. However, PCI bus interfaces are complex, and when implemented in EPLD devices tend to require most of the capacity of the device, even in large EPLDs. Designers interested in integrating both the PCI bus interface and their unique control logic for the back-end device being connected to the bus into the same programmable device will be attracted to the capacity of the higher-density FPGA devices.

The first issue that a designer must face before selecting component technologies is whether full PCI compliance is needed. Full compliance is a requirement for any board intended to be plugged into any PCI system or any system intended to accept any PCI-compatible boards. In other words, full compliance is a must in systems that are to be sold in to the general marketplace and require inter-operability among multiple vendors. However, if the system environment is "closed", in the sense that the PCI bus is being used internally in a system where the equipment designer has control over all devices that interface to that bus (such as an embedded system with no add-in capabilities), then, of course, the designer has freedom to deviate from the specification. This would expand the range of available devices, since many EPLD and FPGA devices are not completely compliant to the PCI specification.

The PCI Special Interest Group (PCI SIG) has published the *PCI Compliance Checklist* of parameters that both system and component suppliers must adhere to in order to claim compliance. It includes a *Component Electrical Checklist* covering areas such as I/O signaling levels, timing specifications, and bus loading. It is important to review this checklist for any device that will connect directly to a PCI bus. By responding appropriately to the items in this check-list, a vendor demonstrates that the minimum amount of work required in order to claim compliance has been completed. Inability to furnish a response to this checklist is sufficient reason to question the vendor's familiarity with the PCI specification and any claims of compliance.

However, while meeting the checklist criteria is necessary, it does not in itself guarantee PCI compliance. Full system-level compliance typically is verified by actually building a board and testing it. To facilitate compliance verification, the PCI SIG conducts workshops where candidate systems and peripherals can be thoroughly tested. The PCI SIG also offers board-level products (some of which use programmable logic) that permit developers to perform their own compliance tests. In addition, simulation models available from Logic Modeling Corp. (a division of Synopsys) have been sanctioned by the PCI SIG for compliance testing.

MEETING PCI PERFORMANCE REQUIREMENTS

The PCI bus supports a maximum transfer rate of 33 MHz over a 32-bit or 64-bit data path (33 MHz transfers of 32-bit data = 132 Mbytes/second). At the maximum 33 MHz clock rate, the 30 ns clock period for a data transfer allocates 11 ns for the output driver, 10 ns for the round-trip bus propagation

Symbol	Parameter	PCI Limit
T_{SU}	Input Set-up Time	7 ns
T_H	Input Hold Time	0 ns
T_{VAL}	Clock to Valid data out	$2 \text{ ns} \leq T_{VAL} \leq 11 \text{ ns}$
T_{ON}	Float to Active delay	$2 \text{ ns} \leq T_{ON} \leq T_{VAL}$

Table 1: Key PCI Timing Parameters

delay, 2 ns for potential clock skew, and 7 ns for input set-up time. This leads to the performance requirements listed in Table 1.

The deterministic timing of most EPLDs makes it a fairly straightforward task to examine the AC timing parameters in the manufacturer's data sheet and determine if that device meets the PCI timing specifications for T_{SU}, T_H, and T_{VAL}. Determining an FPGA's compliance to these specifications is more difficult, since global buffer and interconnect delays as well as logic delays must be taken into account.

Most FPGA data sheets specify clock-to-output timing for registered outputs relative to the clock at the output register (which may reside in an I/O block or nearby logic block). However, the PCI standard specifies pin-to-pin timing, and the T_{VAL} parameter is measured relative to the bus clock entering the FPGA device. Thus, the T_{VAL} clock-to-output valid delay in an FPGA typically would include the delay in inputting the clock signal, bringing it to the FPGA's global clock buffer, routing the clock to the relevant I/O or logic block, the internal clock-to-output delay of the register, and the delay through the output buffer (Figure 1). A clock-to-output valid delay of under 11 ns is difficult to guarantee in large FPGA devices, where it may take 6 to 8 ns to distribute the global clock signal to every flip-flop on the chip.

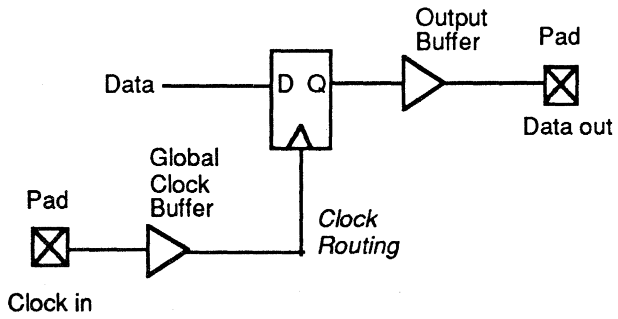


Figure 1: The pin-to-pin clock-to-output timing for an FPGA includes all delays on the clock path from its input pad to the register, and all the delays on the data path from the register to its output pad.

Set-up and hold times for registered inputs also must be examined carefully in FPGA designs. FPGA data sheets typically specify set-up and hold times relative to the internal clock at the register itself, while the PCI standard defines these parameters relative to the clock input pin. Again, the delay involved in inputting the bus clock and routing it through a global clock buffer to the input register and the delay on the data path to the input register also must be taken into account. The internally-specified set-up and hold times must be adjusted to compensate for the delays associated with getting the clock signal to the register. This may result in non-zero hold time requirements in many FPGAs. (Some FPGAs, such as the Xilinx XC3100A and XC4000 families, include a deliberate delay in the data path to an input register to compensate for the clock distribution delay, and, thus, guarantee set-up and hold times relative to the clock input pin. Many other manufacturers do not offer this feature, and the lack of minimum delay specifications on the clock network make the calculation of guaranteed hold times with respect to the clock pin impossible.)

The float-to-active delay (T_{ON}) specifies the time required for an output driver to transition from 3-state to a valid output, timed from a bus clock input. This must occur before a valid output is available (T_{VAL}). This path can involves delays through I/O cells, logic cells, and routing; therefore, this parameter is design-dependent, and requires careful layout along critical paths.

Other, more subtle, performance requirements result from the implementation of the bus interface protocols. For example, the IRDY# and TRDY# signals must respond to a change in FRAME# within one clock cycle. Again, these paths will be design-dependent, involving internal logic and routing paths in the FPGA or EPLD, and require careful attention to the timing of critical paths.

Pipelining

The PCI bus protocols encourage burst-oriented data flows between bus agents, facilitating the use of pipelined data flows within PCI bus interface logic. Pipelining techniques are often key to successfully supporting data transfers at the maximum throughput of the bus. While wait states during bus transactions are permitted and occasionally may be necessary, a high frequency of wait states is counter to the high performance goals of the PCI standard.

The "back-end" of a PCI bus interface typically connects the bus to a processor, memory subsystem, embedded controller, or peripheral device. In most designs, a data buffer, such as a FIFO buffer or dual-port memory, resides between the external device and the PCI bus interface logic, decoupling the speed of the PCI bus from the back-end controller, and allowing multiple data words to be queued for a burst data write to the bus or stored during a burst data read from the bus. Pipelining techniques are then used to control the data flow through the interface logic and onto the bus at maximum speed. This increases the latency of data transfers, as multiple clocks may be required for the first data word to move through the stages of the pipeline and onto the bus, but can allow maximum throughput during a burst transfer by providing new data each bus clock cycle once the pipeline is filled. For example, a two-stage pipeline would require the data be transferred from the back-end device to the first stage on one clock edge, and then driven to the register that drives the bus on the next clock edge. Thus, at the start of a burst read from this agent, a single wait state may be required for the first data byte (while the data is being transferred to the bus interface's register), but a new data word could be read from the agent on each subsequent clock of the burst transfer (assuming that the back-end can supply new data to the pipeline at each clock).

Implementing State Machines

Typically, the signals involved in bus transactions, such as FRAME#, IRDY#, and TRDY#, and the operation of data flow pipelines are controlled by state machines in the bus interface logic; example state machines for controlling bus signaling are provided in Appendix B of the PCI Specification (PCI SIG 1993). To comply with PCI bus protocols and performance requirements, bus control signals must be sensed within the 7 ns input setup time. In some cases, such as initiator-generated wait states during reads, these signals must be responded to on the first clock edge after their activation. Thus, high-performance state machines are required.

For FPGAs, one-hot-encoded (OHE) state machines are recommended (that is, state machines with one register per state and minimal decoding logic). These are well-suited for register-rich FPGA architectures. On the other hand, fully-encoded state machines are better suited to the AND-OR plane logic of EPLDs.

For performance reasons, the transaction protocol state machine often is divided into two levels. The top level sets the data pipeline up for continuous data flow and controls the front end of the cycle. However, while the PCI protocol is compatible with data pipelining, there are two exceptions: the sequence of events that starts on the last transfer of a burst, and when an initiator inserts wait states on a read cycle. Thus, at a second layer, the IRDY# and FRAME# signals must be sensed directly and responded to within their 7 ns set-up time on the bus, so they directly gate state machine outputs that control the appropriate bus control signals.

Parity Generation and Checking

Parity generation and checking is another example where the bus protocol leads to a performance requirement that can be problematic, particularly in FPGA devices, and pipelining often is required. For a 32-bit bus implementation, parity must be maintained for the 36-bit data field consisting of AD[0-31] and C/BE#[0-3] during both address and data cycles. Typically, FPGA logic blocks and EPLD macrocells have a fan-in of less than 36 - far less in the case of FPGAs - so parity generation and checking functions will require more than one level of logic, with the resulting performance implications.

All agents must generate an even parity bit when writing address or data information to the bus. The PAR signal must be driven to the appropriate state and be present on the bus one clock cycle after the address or data. In other words, this signal must be driven out to the bus within 11 ns of the bus clock edge wherein valid address or data information is transferred. Inevitably, signals cannot pass through the multiple logic levels of a parity generation circuit in this short period. Rather than adding wait states to each data transfer to compensate for a slow parity generator, pipelining of the multiple levels of the parity generation circuit can allow full speed operation during burst data transfers. The parity generation pipeline would work in conjunction with the data flow pipelining, with "partial parity" results moving through the pipeline in concert with the data.

For example, consider an EPLD or FPGA capable of implementing six-input functions in each logic cell. A two-stage parity generation pipeline would be required (Figure 2). The first stage of the pipeline generates parity calculations on six 6-bit sub-segments. The parity results of the first stage are registered on the same clock edge in which the back-end data is loaded into the interface register and driven onto the bus. The second pipeline stage then completes the parity calculation; the output of this stage drives the PAR signal. For FPGA devices whose

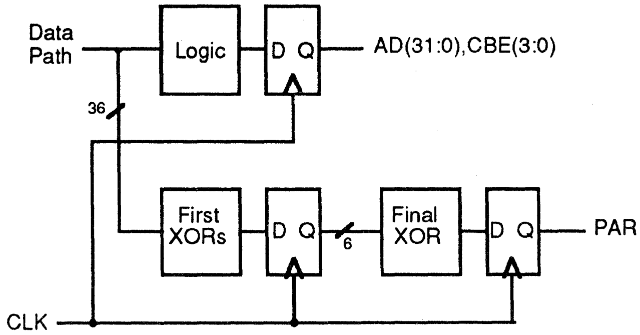


Figure 2: A two-stage pipeline for parity generation

logic blocks have a fan-in less than six, the parity generator may require three-levels of logic. Depending on the set-up time of the data from the back-end to the register driving the bus and the achievable performance of the FPGA device, the first two levels of logic may be implemented in the first pipeline stage, or a three-level pipeline may be required (which may, in turn, be a determining factor in the number of pipeline stages in the data flow.) For initiator agents driving addresses onto the bus, the address and command information must be fed to the parity generation pipeline the requisite number of clocks before the initiator begins the actual bus transaction by asserting FRAME#, in order to insure that the PAR bit is available at the correct time.

Parity checking is not required in target agents, and, even if available, is activated only if a parity enable bit in one of the PCI configuration registers is set. In the event a parity error is detected when reading address or data information from the bus, an active SERR# or PERR# signal must be available on the bus one clock period after PAR. Thus, the error signal must be driven onto the bus within 11 ns after the clock edge where PAR is sensed. The best approach is to generate an even parity bit for the data as it is received and compare it to the PAR signal that arrives one clock after the data. Thus, parity generation can occur during the entire clock cycle following the latching of the data, and the parity bit latched on the next clock edge. (Careful placement and routing may be required to traverse the multiple logic levels of a 36-bit parity generator within 30 ns.) At the clock edge where PAR is sensed, the generated parity bit must then be XORed with the PAR signal, ANDed with the enable bit from the configuration register, and drive the SERR# or PERR# signal, all within the next 11 ns. This implies that the sum of the clock-to-out delay of the register holding the generated parity, the delay through one level of logic, the output buffer delay, and all associated internal routing delays along this path must not exceed 11 ns. While possible in the highest speed devices, this can be a difficult challenge in any technology, and discourages designers of many target-only devices from including parity check capability.

Similarly, wide decode logic is required by potential bus targets to decode bus addresses, and is likely to involve multiple levels of logic within an FPGA or EPLD. However, the PCI standard allows up to four clock periods for an addressed device to activate its DEVSEL# signal; this is ample time to traverse several levels of logic in today's programmable logic devices.

I/O REQUIREMENTS

The PCI bus is unterminated, operating on the principle of reflective wave signaling. The initial output signal, therefore, has half amplitude. It travels to the end of the non-terminated bus, and gets reflected back towards the source to become a full-amplitude signal. This scheme demands strict control of device drive characteristics. The driver must have an output impedance in both the High and Low states that is roughly the same as the characteristic impedance of the driven bus. When that condition is met, the returning signal is absorbed without any further ringing or reflection. In order to achieve 33 MHz, the round-trip delay must be limited to 10 ns, limiting the physical length and allowable capacitive loading of the bus.

Thus, unlike traditional bus specifications, PCI defines AC switching characteristics as well as DC parametrics. In other words, as well as specifying I/O sink and source limits at logic 0 and 1, PCI also includes specifications for sink and source switching currents across the transition from one logic level to another. These are specified as regions and specific points on a current vs. voltage graph (or "I/V curve") within the PCI Specification. This type of I/O drive information typically is not readily available in FPGA and EPLD data sheets, and may need to be obtained from the manufacturer. Again, designers are encouraged to examine device manufacturers' responses to the items in the PCI SIG's *Component Electrical Checklist*.

Bus loading must be strictly controlled in order to maintain performance. Each add-in card edge-connector finger can attach to only one device pin of no more than 10 pF, except for the CLOCK pin which can be 12 pF. Programmable logic devices in plastic packages typically meet this requirement; although data sheets often list higher input capacitances, these values usually only apply to ceramic packages, where the multi-layer package increases pin capacitance.

The SERR# and interrupt request bus signals must be implemented as "open drain" outputs that can be shared by multiple agents in a wired-OR manner. While most FPGAs and EPLDs do not incorporate true open drain output structures, the functionality of an open drain output is easily implemented in an FPGA or EPLD with three-statable output buffers. This is accomplished by supplying a zero to the data input of the buffer and driving the buffer's output enable input instead of the data input.

Some FPGA and EPLD devices have restrictions as to the number of different signals that can be used as output enables for the device's output buffers. These devices generally are not good candidates for PCI bus interface designs. Since different bus control signals must be driven at different times and most control signals are three-state signals that must be driven by three-state buffers (and separate control of the output enable is needed for each of the "open drain" signals, as described above), PCI interface designs require significant flexibility in the generation of multiple output enable signals.

Pin Placement

The PCI standard specifies a maximum trace length of 1.5 inches from the card edge connector to the PCI device for all 32-bit signals in order to limit trace capacitance. In all, a minimum of 47 pins are required for a target-only device and 49 for an initiator. These requirements dictate the use of a high pin count device in a small, dense package, such as the popular plastic quad flat pack (PQFP) package.

The PCI specification recommends a pinout for PCI interfaces in quad flat pack packages that is designed to align with the board's edge connector pin assignment. However, other factors that could influence pinout choices include simultaneous switching and board layout considerations, and internal logic placement along critical paths within the FPGA or EPLD. To avoid switching noise problems, address/data bus signals often are placed in groups surrounding package ground pins. The target FPGA or EPLD device should provide adequate routing near its I/O cells to provide flexibility in I/O placement while allowing complete, fully-routable designs.

OTHER ARCHITECTURAL CONSIDERATIONS

The PCI bus is a synchronous bus based on a single master clock signal; this implies that bus signals should be registered as they exit and enter bus agents. Thus, PCI interfaces demand an architecture with adequate register space for latching bus signals as well as generating state machines, pipelines, and other internal logic. High-density, register-rich FPGAs easily fulfill this requirement. EPLDs typically provide only one register per macrocell; however, some EPLDs, such as the Xilinx XC7300 family devices, feature additional, independent registers in their input paths, as well as the macrocell registers. These structures are ideal for latching bus inputs, and are a necessary resource for meeting the capacity and density requirements of PCI bus interface design.

PCI devices must implement a basic set of configuration registers, divided into a predefined, 64-byte header region and a 192-byte device-dependent region. Many of these registers or fields within these registers are optional. For EPLD-based interfaces, these registers would be implemented in an external memory device. FPGA-based designs may include some or all of the configuration bits either within the FPGA or in external memory devices, dependent on the design requirements and FPGA capacity, .

The PCI standard includes specifications for both 5 V and 3.3 V signaling environments; PCI subsystems can be 5 V only, 3.3 V only, or universal (both). A keyed connector scheme prevents damage to single voltage cards. Currently, FPGA-based PCI interfaces must be 5 V only designs; the 3.3 V FPGAs available today do not provide adequate performance for this application. Some EPLDs, such as the XC7300 family, while running at 5 V internally to provide the required performance, can accommodate either 3.3 V or 5 V signal levels on their I/O pins. Thus, these devices can be used on universal cards that provide both signaling environments.

DEVELOPMENT SYSTEM CAPABILITIES

An often overlooked factor when evaluating programmable logic devices for possible use in PCI systems is the capabilities of their development tools. Since the performance requirements of the PCI standard tax the capabilities of most FPGA and EPLD devices, development tools capable of easily producing high-performance layouts are mandatory. If a high-level hardware design language is used for design entry, logic synthesis tools that are optimized for the target architecture are required. While automated layout tools should be robust, the intricacies of PCI design may require that the designer exercise some control over the tools, especially in the placement and routing of critical paths and the placement of I/O pins. Thus, some form of placement control is needed, and floorplanning support is desirable. For FPGA-based designs, timing-driven place and route tools such as XACT-Performance™ from Xilinx and TimingWizard™ from NeoCAD can ease the design process by allowing the specification of target performance requirements for entire paths through the design. "Re-entrant" FPGA place and route tools, wherein the placement and routing of a previous version of a design can guide the implementation of a new version with minimal changes, can greatly ease the design process.

EXAMPLE DEVICES AND DESIGNS

Examples of fully compliant programmable logic devices include the XC3100A FPGA and XC7300 EPLD families from Xilinx Inc. *PCI Compliance Checklist* data has been submitted to the PCI SIG (and is available to interested users) for the -2 speed grade of the XC3100A family, and -10 and -7 speed grades of the XC7300 family.

Several PCI designs have been based on the XC3100A FPGA family. In fact, the PCI SIG chose an XC3100A device for the board developed for use in their BIOS compliance test kit. A target interface design that links the PCI bus to a slave processor through a dual-port RAM has been incorporated in an XC3164A-2 device in a 160-pin PQFP package (Figure 3). The design was coded in Verilog HDL, synthesized using Exemplar Logic's CORE™ tools, and verified on a PC using Simucad's Silos/Verilog simulator. This design utilizes only 40% of the logic blocks available in the XC3164A FPGA. Design files and an application note describing the design are available from Xilinx (Xilinx 1994a).

Likewise, an application note and design files are available for a PCI target interface implemented using two EPLD devices: an XC73108 and an XC7354 (Xilinx 1994b). This design also can be collapsed into a single XC73144 device. The "back-end" interface is a FIFO buffer connected to a DRAM memory subsystem with its own memory controller.

While not fully compliant, the XC4000-4 FPGA family also has been used in a number of "embedded system" PCI bus implementations. (The XC4000-4 FPGA devices fall just short of meeting the T_{VAL}, T_{SU}, and T_H timing requirements, but are compliant in all other aspects.) However, XC4400 HardWire devices, mask-programmed versions of the XC4000 devices, are fully-compliant, allowing for prototype development with the programmable version, but high-volume manufacturing with the fully-compliant HardWire version. A higher-performance version of the XC4000 FPGA architecture will be available in the second half of 1995, and is expected to be fully compliant. The XC4000 architecture has several features that facilitate PCI bus interface design, including the ability to implement 9-input functions in a single block (easing parity generation and checking), and on-chip RAM capability (facilitating the on-chip implementation of the PCI configuration registers).

SUMMARY

While careful design is required, PCI-compatible EPLD and FPGA devices bring the system-integration, flexibility, and time-to-market benefits of high-density programmable logic to the PCI design community. These devices can provide the performance, density, and routability to handle complex structures such as pipelined data paths, 32-bit parity generation, and PCI bus control.

REFERENCES

1. PCI SIG, PCI Local Bus Specification Revision 2.0, April, 1993

2. Xilinx Inc. application note, Fully Compliant PCI Interface in an XC3164A-2, 1994

3. Xilinx Inc. application note, Designing Flexible PCI Interfaces with Xilinx EPLDs, Document No. 0010216-01, 1994

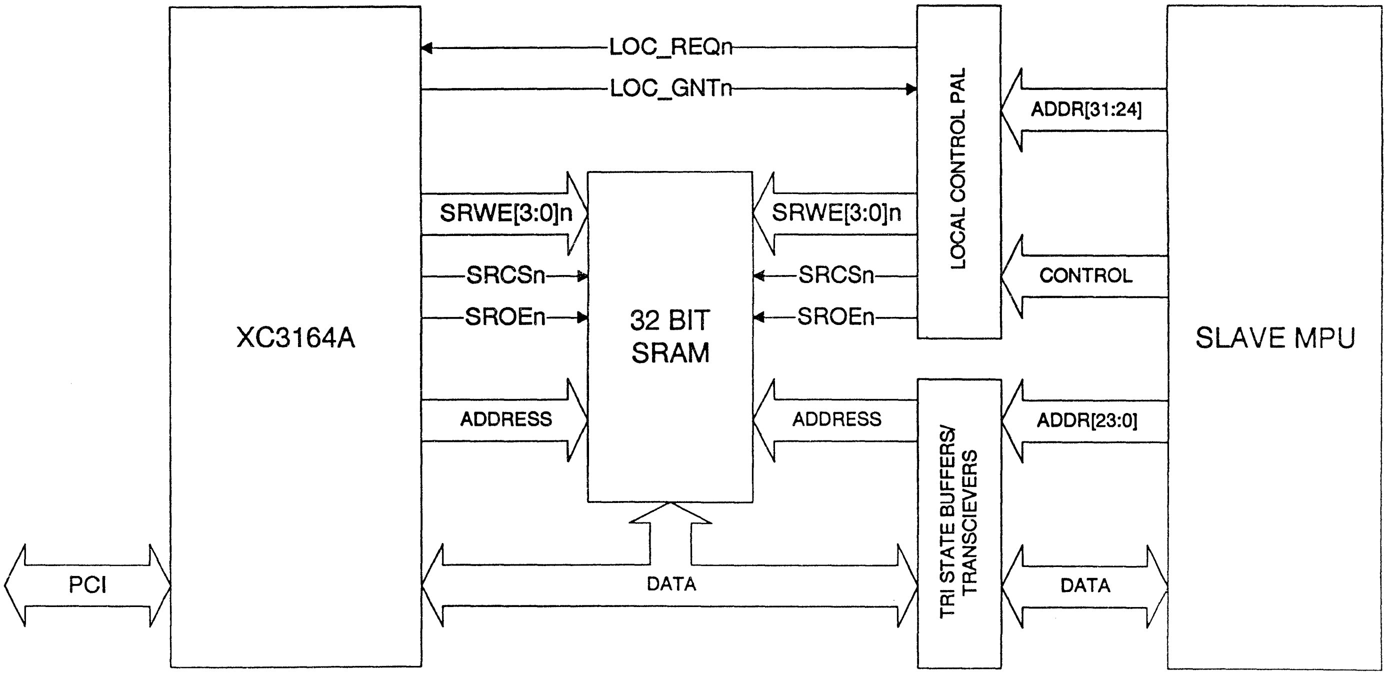


Figure 3: Block diagram of system employing the XC3164A FPGA for a PCI bus interface. The back-end of the interface is a bank of static RAM shared by an on-board host microprocessor.

TRENDS IN SYSTEM LOGIC DESIGN

Wen-Chi Chen
VIA Technologies, Inc.
5020 Brandin Court
Fremont, CA 94538 USA

ABSTRACT

In the past ten years, PC system logic has evolved from simple glue logic plus DMA, interrupt control and timer functions into much more powerful and sophisticated system control functions.The general trend has been moving toward

- Higher performance CPUs
- Higher frequency CPU buses
- Bigger and faster cache and DRAM
- Faster and better I/O buses
- Advanced features such as "Green" and "Plug and Play"
- Higher integration

System logic can be divided into several subsystems namely, CPU interface, memory control, bus interface and other features.

CPU INTERFACE

From the original 8088 to today's high end Pentium and equivalents, CPU performance has grown at a rate of 50% per year. In the future, system logic designs will need to support new CPU features such as write-back cache and Green functions. The CPUs of the various vendors differ today, and in the future they will differ even more. System logic designs will try to support as many of the differing CPUs as possible. Most system logic chip sets will support Pentium class CPUs, such as AMD's K5 and Cyrix's M1, but they will not support NextGen's CPUs due to their different cache architecture.

Dual, or multiple processor support, will be one of the major issues in future CPU interface designs.

MEMORY CONTROL

Direct mapped cache, both asynchronous and synchronous, is generally supported by system logic designs today. Originally the size of cache for 386 systems was 32K Bytes to 64K Bytes. Pentium system logic designs now support up to 2 Megabytes, and in the future most designs will support synchronous pipelined 3.3v cache. Second level cache may move inside P6 type CPUs, and consequently system logic designs will support either much bigger cache, or no cache at all.

DRAM interfaces will go through even greater change. EDO (Extended Data Out) DRAM support will be a requirement. But other DRAM such as synchronous DRAM, EDRAM, RAMBUS and others may also be supported by system logic. 3.3v DRAM is used mostly for mobile systems now, but it may become popular in desktop systems too.

BUS INTERFACE

The ISA bus will remain long after the EISA and VL buses are dead, and the PCI bus will be supported by all system logic designs. Its transfer rate will become an important consideration. Integrated buffering will be common, and also 64-bit PCI buses will be more common. The popularity of 66 MHz PCI buses will depend upon reliability issues. PCI to PCI bridges and CardBuses will become key features.

OTHER FEATURES

"Green" functions and "Plug and Play" will be supported by all the major system logic designs. Other functions, such as the keyboard controller, the enhanced IDE controller and the real time clock will be integrated. The latest trend is to support Native Signal Processing (NSP) which may lead to the integration of audio, modem/fax and certain video functions. Other possibilities are to integrate "Super I/O", gaphics control and LAN functions.in system logic designs

COST CONCERNS

Cost is always a major consideration for system logic designers in attempting to increase performance by adopting advanced features. Most cost decisions involve

- Pin count: 100, 160, 208, 240, 304 and above
- Package: PQFP, or DGA
- Gate count
- Process technology

New PC Card ICs Contribute to the Proliferation of the PCI Bus

Mark Bode, David Dickens & Tony Wutka
Texas Instruments -- PCIbus Solutions
P.O. Box 84, M/S 814
Sherman, TX 75090

ABSTRACT

The emerging PCI standard brings with it the need for unique, innovative solutions for connecting system components to it. The advent of PCI requires devices which can bridge between PCI and other peripherals, such as PC Cards. Systems also must be able to support legacy software and hardware. Issues such as legacy DMA and standard ISA interrupt handling are difficult to implement under the current PCI specification. Creative solutions exist for PC Cards that address both legacy DMA (Direct Memory Access) and standard ISA interrupt handling in a PCI environment.

INTRODUCTION

After years of experimenting, the PC industry is converging on a single high-speed local bus standard for motherboards. This standard, the Peripheral Component Interconnect (PCI) bus, is rapidly becoming the local bus of choice in both desktop and portable PCs. To meet the emerging demand for devices that connect peripherals to the PCI bus, Texas Instruments is developing a family of PCI bus interface ICs.

PCI's high performance, scalability, microprocessor independence and endorsement by the PC industry ensure that the bus standard will become dominant in the market during the next few years. TI estimates that by 1997 the market for PCs using PCI will have grown to nearly 40 million units a year, more than 60 percent of the entire PC market.

In addition to the high growth of the PCI industry, other peripheral industries such as PC Cards are also experiencing rapid growth. CardBus, an extension to the new PC Card specification, is fully backward-compatible with the PCMCIA 2.1 PC Card specification. CardBus provides a relatively easy upgrade path from R2 (PCMCIA 2.1) PC cards. CardBus provides added features over R2 cards, such as bus mastering capabilities, 32-bit data paths, 33 MHz speed and future lower voltages. CardBus controllers being designed by Texas Instruments will be among the industries first to interface PCI to CardBus. These controllers also provide innovative solutions which can help simplify support of legacy software and hardware.

New PC Card controllers will address some of the weaknesses of the current PCI specification. The first of these is the lack of an easy to implement standard for handling legacy DMA. A unique method of resolving legacy DMA issues is to use a distributed DMA scheme. The second is the difficulty and overhead of supporting standard ISA interrupts in a PCI environment. The preferred method of handling standard ISA interrupts is to utilize a serial interrupt design.

PCI: THE PC BUS OF THE FUTURE

PC manufacturers have long looked for a high-speed alternative to the ISA bus, which dates from the original IBM XT and AT designs. The list of alternatives reads like alphabet soup: EISA, MicroChannel, VME, VME64, FutureBus+, VL-bus. While all of these alternative buses have advantages, none is as well designed and universally accepted as PCI to take PCs into the future.

PCI was developed by the PCI Special Interest Group (SIG), an industry-wide committee representing more than 300 companies. The PCI Local Bus Specification, revision 2.0, approved in April 1993, is the latest version of the standard.

With a 32- or 64-bit multiplexed data and address path, PCI operates at frequencies of up to 33 MHz and can support up to 10 loads per bus. Although PCI was created as a local bus, it also has I/O capabilities that make it advantageous for exchanging massive amounts of data with peripherals such as hard disk drives and LANs.

PCI is platform-independent, supporting both CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer) microprocessors. It is also backward-compatible with earlier x86 processors and software -- always a key factor to acceptance in the PC industry.

PCI's high bandwidth of 132 Mbytes/s for 32-bit transfers and 264 Mbytes/s for 64-bit transfers means that it can support the mixed video, audio and other data transfers necessary for multimedia.

Because of the multiplexed data and address lines, PCI cards require few signals -- just 49 for bus masters and 47 for slaves. Having fewer signals means that the bus cuts down on pin count, board space and layers, saving expense for motherboard manufacturers. PCI specifies both 3.3-V and 5-V signal levels, so it easily transitions to the low-voltage systems that are rapidly entering the market today.

PCI's scalability allows it to be used in a variety of systems, ranging from notebooks to desktop PCs to workstations. The ubiquity of PCI will also serve to drive down costs, making it even more attractive for computer manufacturers.

PCMCIA: THE KEY TO MOBILITY

As a local bus, PCI handles high-frequency traffic among the CPU, memory and controllers for high-speed devices such as video displays. Although PCI has I/O capability itself, it also needs to interface to other I/O buses. Among these, the most important emerging standard is the PC Card (commonly called "PCMCIA") bus and its extension CardBus.

The Personal Computer Memory Card International Association (PCMCIA) in association with Japan Electronic Industry Development Association (JEIDA) has worldwide support from more than 500 member companies for its PC Card standard. PCMCIA revision 2.1, the current version of the 16-bit specification, represents the culmination of various improvements to earlier releases of memory and I/O cards for PCs.

PC Cards support typical data transfers of 5 to 10 Mbytes/s and a theoretical limit of 20 Mbytes/s. In addition, with automatic system configuration, a PC Card is plug-and-playable through hot card insertion even after the system has been powered on.

The PC Card standard brings several advantages to small form-factor add-ins. Its 3.3- and 5-V signaling options support various schemes of system power management, helping to reduce power consumption and prolong battery life. The PC Card standard also supports the miniaturization of PC products for lighter weight and greater mobility.

These same advantages are also offered by a variant of PCI, Small Form-Factor PCI. While there is some overlap in application between the two interfaces, Small Form-Factor PCI is finding more use in embedded systems, while PC Cards are more widely used for add-in peripheral cards.

An improved version of the PC Card standard has been announced by PCMCIA and JEIDA. This new standard will support a multiplexed 32-bit address/data path defined for CardBus PC Cards, while still maintaining compatibility with 16-bit version 2.1 PC Cards. With a wider data path and operating speeds of up to 33 MHz, CardBus cards will support data transfer rates up to 132 Mbytes/s. In addition, CardBus supports bus mastering capability, operates at 3.3V and allows for the next generation of lower device operating voltages. With these enhancements, CardBus will further enhance the mobility of future PC systems.

DISTRIBUTED DMA: THE PREFERRED SOLUTION

A standard feature of the PC-AT ISA architecture is the 8237 DMA(Direct Memory Access) Controller. The dual cascaded 8237 configuration provides seven DMA channels, by which application software may program memory-to-I/O or I/O-to-memory transfers without processor involvement. DMA transfers are useful in transferring blocks of data from peripherals to main memory without taxing the processor itself. Such transfers are common in floppy disk drives, parallel ports, sound cards, and other I/O devices.

The evolution of the PC architecture has seen the introduction of several new peripheral bus protocols which define new ways of connecting current peripherals and open the door to new ones; PCI and PC Card are perfect examples. However the support of legacy DMA peripherals and the applications which use them require a method of implementing DMA in a PCI and/or PC Card environment. This need prompted PCMCIA to provide DMA support in the upcoming release of the PC Card standard. The lack of support in the PCI environment prompted a group of computer and semiconductor makers to define DMA support for PCI. The result is a distributed DMA scheme for PCI.

The distributed DMA concept dictates that each PCI device implements as many slave DMA channels as required by the peripheral. For Texas Instruments CardBus controller, the PCI1130, this means that one slave DMA channel per socket, or two channels total, are supported. In order to maintain the legacy DMA programming model, one DMA device in the system is designated the master DMA device. This device will claim PCI I/O reads and writes to the legacy DMA control registers. The master DMA device relays the information written to these legacy DMA addresses to the proper slave DMA device, such as the PCI1130.

Each slave DMA channel has a set of registers mapped into a 16-byte window in PCI I/O space. The location of this window is specified by the Slave DMA Configuration Register, found in the slave device's PCI Configuration space. Two such registers may be found in the PCI1130.

By programming the Slave DMA Configuration Register, the master DMA device may communicate with the slave DMA. After these registers have been programmed by the master DMA device, the PCI1130 may respond to a PC Card's assertion of DREQ by bus mastering on the PCI bus and transferring the required data.

A distributed DMA scheme offers several benefits over a concentrated DMA solution. First, by implementing the scheme mostly in hardware, the legacy DMA programming model is retained. This ensures that software written to this model will now function as intended in a PCI system. Second, the distributed DMA scheme 'distributes' the burden of DMA support among all of the devices which will support it, rather than requiring a single device to implement the entire dual-8237 functionality. A drawback to the concentrated solution is that each DMA transfer from memory-to-I/O, would require two PCI transactions: one from the target to the master, and one from the master to the destination. The distributed approach programs the slave DMA device to carry out the transfer directly.

An attractive byproduct of the distributed DMA scheme described here is that PCI devices supporting this scheme will automatically support a mechanism for PCI bus mastering. New application software or device drivers may program the Slave DMA registers directly without being constrained by the legacy DMA programming model. This allows new applications to exploit the bandwidth of the PCI bus by supporting 32-bit transfers over the full PCI address range.

Serialized Interrupts: the future alternative

Another standard feature of the PC-AT architecture is the 8259 Programmable Interrupt Controller. Traditionally, there are fifteen interrupts available in a PC-AT system. The interrupt controller responds to requests for help or service by executing interrupt service routines.

An alternative method to standard ISA interrupts is to implement shared interrupts, as defined by the PCI specification. Shared interrupts offer the system designer a simplified means of handling interrupts, but to date have proved difficult to implement. To build a PCI system and maintain backward compatibility with legacy software, a system must be capable of responding to standard ISA interrupts.

A superior method of handing ISA standard interrupts is to provide them serially to the interrupt controller for servicing. This method was defined by several of the industries leading suppliers of systems, system controllers and peripheral components.

Since the interrupt stream is constructed utilizing a wired-OR scheme, IRQSER, a common pin, is driven low for each corresponding IRQx on its assigned clock cycle and allows for their replication at the host controller. This method has the advantage of reducing the number of lines required to service standard ISA interrupts. Using a device such as the Texas Instruments PCI1130 CardBus controller, the number of board traces required to service standard ISA interrupts can be significantly reduced.

Bus Interface ICs: the cornerstones of PCI

Because of the growing significance of PCI, TI is supplying interfaces that will bridge between PCI and other system functions, such as PC Cards (PCMCIA) and CardBus. The company is well-positioned to play a role in enhancing the viability of PCI. As an active member of both PCI SIG and PCMCIA, TI plays a part in defining these standards and has an early perspective on future developments.

In addition, TI's core competencies assure its customers that it can supply devices in the volumes needed for a rapidly growing PCI bus market. Among these competencies is TI's experience in designing, manufacturing and marketing mixed-voltage (3.3-/5-V) ICs.

PCI is the preferred bus for new PC designs. As the market for PCI continues to grow, PC manufacturers will need off-the-shelf components to help them integrate the bus. With its manufacturing strength and low-voltage leadership, TI is positioned to provide these essential components for PC systems. PCI bus interfaces from TI will be key to the future of PC systems.

Number Nine's View on PC Graphics -- Now and into the Future

Presenter: Andrew Najda
President and Co-founder
Number Nine Visual Technology Corporation

Number Nine Computer Corporation, renamed Number Nine Visual Technology Corporation to reflect its product direction in the graphics marketplace, has been a leading provider of PC-based high-performance graphic display solutions for more than a decade. In the recently released International Data Corporation report entitled "The Intel VGA Add-In Board Market", IDC ranked Number Nine as an industry leader in high-end VRAM-based graphics and #4 in the overall market.

Founded in 1982, Number Nine has consistently delivered preemptive market-driven technology. Number Nine delivered a number of world's firsts, starting with the first 256- and 16.8- million-color cards for the PC. The company went on to develop the first graphics accelerator board with a built-in processor, the first solution to allow both the host CPU and the graphics processor to draw in parallel, and the first graphics productivity and utility software for Windows users. Only months after Number Nine began shipping a new series of 64-bit graphics accelerator boards, Number Nine again demonstrated its industry leadership by introducing Imagine128, the worlds first 128-bit graphics accelerator chip and board family. Number Nine is continuing this tradition with the development of Imagine128 Pro. This product will incorporate 8MB of high-speed VRAM, a #9 Imagine128 processor, a 128-bit data path from the processor to memory, and a 128-bit DAC (digital to analog converter). This should be the first commercially available graphics accelerator board to utilize a 128-bit DAC.

Computer manufacturers are in a race to introduce faster and more powerful systems. The line that separates the PC from the workstation is becoming increasingly difficult to define as PC performance continues to increase and prices fall. Computer manufacturers are consistently turning to graphics board manufacturers for assistance in improving overall graphics performance, added utilities and increased functionality. It is this added value that helps the computer vendor differentiate their system from the competitor.

Andrew Najda, president and co-founder of Number Nine will be present to speak on events that are currently effecting the graphics industry. Additionally, Mr. Najda will discuss where the industry is heading as a whole. Topics being covered will be:

1) How newer 32-bit operating systems will increase graphics demands on the overall system.
2) The strengths and weakness of current and new graphics memory technologies.
3) The integration of graphics, video and audio.
4) How 3D will evolve at the low-end for entertainment markets, to the more demanding requirements of high end engineering applications.

This session promises to be very educational for all those that attend.

PCI Host Bridge for the 60x Family of PowerPC™ Microprocessors

Christopher D. Bryant, Michael J. Garcia, Laura A. Weber
Motorola Incorporated
6501 William Cannon Drive West
Austin, Texas 78735-8598

ABSTRACT

This paper describes a single chip Peripheral Component Interface (PCI) host bridge for the PowerPC 60x family of microprocessors. The MPC105 is a high performance, low power, low cost chip which integrates all functions of a PCI host bridge, a memory controller, and secondary L2 cache controller. This solution provides system designers the ability to design a wide range of performance systems based on the PCI bus and the PowerPC architecture.

ARCHITECTURAL OVERVIEW

The MPC105 is partitioned into four interfaces, the processor interface, the second level (L2) cache interface, the memory interface, and the PCI interface. A central control unit provides arbitration and coherency control between each of the interfaces. This central control unit supports concurrent operations on the processor/memory bus and the PCI bus, allowing transactions such as a processor write to memory to occur while a master on the PCI bus is writing to memory via internal buffers. The processor interface is a high bandwidth, high performance, TTL compatible interface which supports any of the MPC60x PowerPC microprocessors and a second level (L2) cache or two MPC60x PowerPC microprocessors. The memory interface is highly flexible, supporting either DRAM or SDRAM in sizes up to one gigabyte, and in up to eight banks. The PCI interface is fully compliant with the PCI Local Bus Specification Revision 2.0[1] and all its supplements and functions as both a master and target device.

The MPC105 connects directly with the PCI bus and the processor bus, and shares the data bus to system memory with the processor. The processor/memory bus and the PCI bus are synchronized with the use of a Phase-locked loop (PLL) clock design. An H-tree clock distribution network is used to minimize the clock skew to less than 500ps across the chip. The MPC105 supports PCI bus operations at a frequency between 20Mhz and 33Mhz, with the processor/memory bus running at either the same frequency or two times the PCI frequency. A typical system using the MPC105 is shown in Figure 1.

The following sections will describe the functionality and capability of each of the interfaces, the central control unit, and various other features such as power management modes, error detection and reporting, and chip technology.

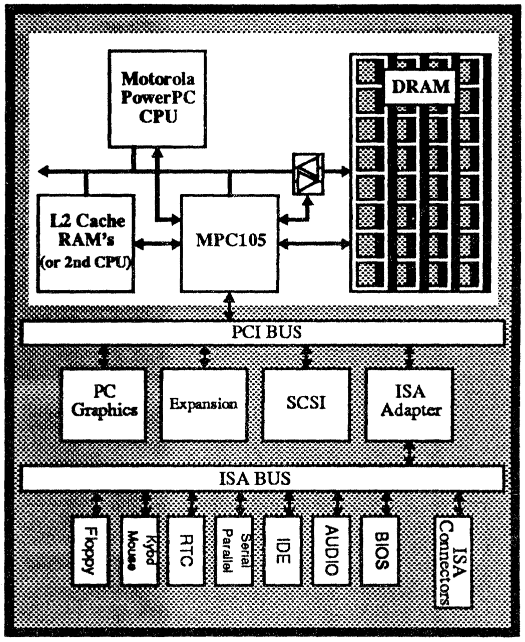


Figure 1 **Typical PC system using the MPC105.**

PCI INTERFACE

The PCI interface is fully compliant with the PCI Local Bus Specification Revision 2.0[1] and all its supplements and functions as both a master and target device. The interface supports PCI bus speeds of 0Mhz in the sleep or suspend power savings mode and a range of 20Mhz to 33Mhz in the full on mode. The interface can be programmed for either Little Endian or Big Endian formatted data. It provides the data swapping, byte enable swapping, and address translation in hardware. The interface supports parity checking and error reporting as both a master and a target.

The interface supports memory reads and writes, I/O reads and writes, configuration reads and writes, special cycles and interrupt acknowledge as a master. As a target the interface supports memory reads (including memory read line and memory read multiple), and memory writes (including memory write and invalidate).

The interface is controlled by a master and a target state machine running independently of each other. This allows the MPC105 to run two separate transactions simultaneously. For example, if the master is trying to run a burst write to a PCI device, it may get disconnected before finishing the transaction.

If another PCI device is granted the PCI bus and requests a burst read from system memory, the interface can accept the burst transfer and continue the burst write when next granted the PCI bus.

The MPC105 as a Master

Upon the detection of a valid command from the central control unit, the PCI interface requests the use of the PCI bus if not already granted. Once granted, the MPC105 drives a full 32-bit address and command. The master interface supports reads and writes of 1, 2, 3, 4, or 32 bytes without master initiated wait states. The one, two or three byte transfers can either be aligned or unaligned. The four and thirty-two byte transfers must be aligned. The master part of the interface does not run fast back-to-back or interlocked access. The master interface does support decode for all 21 identification selects, any of the various device selection timings, master abort, target abort, target retry, and target disconnects.

The MPC105 as a Target

As a target, upon detection of an address phase the interface simultaneously decodes the address and command to determine if the transaction is for system memory. If the transaction is destined for system memory the interface latches the address and decoded command and forwards them to the central control unit. On writes to system memory data is forwarded along with its byte enables to the central control unit. On reads 4 bytes of data are provided to the PCI bus regardless of the byte enables.

The target supports both PCI compliant fast back-to-back transactions, interlocked accesses using the PCI lock protocol, target-abort and target retry, The MPC105 uses the fastest device selection timing and can accept bursts writes of up to 32-bytes with no wait states. Burst reads of up to 32-bytes are also accepted with wait states inserted depending upon system memory speed. The target interface will disconnect at the end of a cache line (32-bytes) to force a new address for snooping purposes.

Design Issues

Most differences in the operation of the processor bus and the PCI bus are resolved within the PCI interface. For example, if the processor bus is operating at twice the frequency of the PCI bus, the internal control of the MPC105 operates at the processor bus speed and the PCI interface synchronizes transfers with the slower PCI bus.

Another design issue requiring special attention was the ability to interface between the 64-bit processor data bus and the 32-bit PCI bus. In this case, the interface to the central control unit is 64-bits wide. For PCI writes to memory, the interface latches two 32-bit beats of data and 4-bit byte enables. The interface then forwards all 64-bits of data and 8 byte enables to the central control unit. If an odd number of PCI data transfers is done, the data written to the central control unit and to system memory is still 64-bits with the proper byte enables.

On PCI reads from system memory, the central control unit forwards 64-bits of data to the target interface. The interface then selects which thirty-two bits of data to send out onto the PCI bus. The master part of the interface has similar pointers for selecting 32-bits out of 64-bits for processor writes to PCI.

MEMORY INTERFACE

DRAM Support.

The memory interface of the MPC105 controls transactions to and from system memory and supports a maximum of one gigabyte of DRAM or JEDEC compliant SDRAM. The memory interface's flexible design supports a variety DRAM configurations through SIMM's and/or direct board attachments. Support for up to eight banks of memory is provided through the use of eight row address strobe lines and eight column address strobe lines, allowing for byte selection during writes. Twelve address pins allow each of the eight banks to be populated with memories from 1 megabit up to sixteen megabits in depth. Memories can be from one to seventy-two bits in width. All banks must be populated with the same type of DRAM, as mixing of DRAM and SDRAM is not supported. It is not necessary to use identical memory chips in each of the eight banks. Individual banks may be of differing size. The memory interface can be configured to provide nine to twelve row bits to a bank, and nine to twelve column bits. The row and column bits are multiplexed onto the twelve address pins. The start and ending addresses for each bank are programmable, allowing appropriate row and column multiplexing.

Timing variables for read and write transactions are controlled through programmable registers, allowing system designers to optimize the MPC105 for a variety of memory designs. Some of these programmable variables include the RAS and CAS precharge times, the RAS to CAS delay time, and the first access CAS pulse width. For SDRAM systems some of the programmable timings include the data latency from read commands, the interval between refresh command to active command, the interval between read and write commands to active commands, and the intervals for active to precharge and precharge to active.

The MPC105 can be configured to provide parity checking. If enabled, parity will be checked for all memory reads and will be generated for PCI writes to memory, L1 or L2 copybacks, and L2 castouts. The processor provides parity for all other processor to memory related transactions.

CAS before RAS (CBR) refresh is used to maintain memory integrity. This interval is programmable with a resolution of one processor bus clock cycle. The MPC105 distributes refreshes to (S)DRAM according to the interval programmed and will bank-stagger the refreshes to minimize instantaneous current consumption. While in power savings mode or during system shutdown, the MPC105 can be programmed to use normal CBR refreshes, self refresh mode (for memories that support this functionality), or CBR refreshes based on a clock frequency other than the internal clock and supplied to the MPC105 (not supported with SDRAM). Refreshing can be disabled to allow for systems who prefer to copy back main memory to non volatile memories or maintain memory through other means.

ROM Support

The MPC105 supports a 32 or 64 bit wide ROM or an 8 bit wide FLASH ROM with memory sizes for up to sixteen megabytes of ROM or one megabyte of FLASH ROM. The ROM memory can be located off the memory interface or out on the PCI bus. Twenty bits of address and two chip selects, which can be used as bank selects, are provided for systems that are using 32 or 64-bit wide ROM. The MPC105 provides for programmable access timing for ROM so that systems of various clock frequencies may be implemented. The MPC105 can be programmed to support the burst capability available with some ROM memories, taking advantage of access time improvements. The programmable parameters for ROM access have a granularity of one processor bus clock cycle.

For systems that prefer to use an 8 bit wide ROM or flash ROM, 20 bits of address, a chip select pin, a write enable pin, and an output enable pin are provided to ease read accesses and write accesses to flash ROM. The MPC105 only supports Byte write accesses to flash ROM. To reduce bus traffic, individual bytes are read from the byte wide ROM and are gathered in the MPC105 before sending the requested size back to the processor.

PROCESSOR AND L2 INTERFACE

The MPC105 processor interface supports a subset of the PowerPC microprocessor bus capabilities. The subset includes but is not limited to single-beat (8 bytes or less) transfers, burst (32 bytes) transfers, the 60x Little Endian (LE) mode, the 60x 32-bit mode, the address retry mechanism, and pipelined transactions. It uses a 32-bit address bus that is decoupled from the 64-bit data bus, and provides arbitration to these buses for one processor without L2, one processor with look-aside L2 or two processors. It also uses the snoop protocol for PCI to system memory transfers.

The MPC105 processor's interface, upon the detection of a transfer start from the processor, latches the address, transfer type, and transfer size. The interface decodes the transaction to determine whether the transaction is a read, write, or address only and whether the transaction is destined for PCI memory space, PCI I/O space, system memory, ROM or the internal configuration registers. The address and the decoded information are passed along to the central control unit where it is used for internal snooping operations from either the processor or PCI interface. The data phase of the transaction proceeds when it is determined that the transaction will not get retried and when it is the highest priority of the outstanding transactions within the MPC105. Once the data phase has begun, the MPC105 allows pipelining to occur. A one-level pipeline is enforced by extending the address phase of a pipelined transaction until the data phase of the previous transaction is complete. This allows the MPC105 to begin the decode for the next transaction.

The MPC105 provides control for a direct mapped look-aside L2 cache. This L2 cache can be of size 256 kilobytes, 512 kilobytes or 1 megabyte, programmed in either write-through or write-back mode and provides up to 4 gigabytes of cacheable address space. The L2 interface can be programmed to support either asynchronous, synchronous or pipelined SRAM's of various speeds. The programmability of the L2 interface allows flexibility in the choice of SRAM's and Tag RAM's for various processor bus frequencies. The support for L2 cache flush and L2 cache invalidate is provided in hardware.

To determine the L2 response to processor or PCI initiated transactions a hit and modified signal are used along with the address retry, and page protection bits from the processor. The L2 cache will supply data on read hits that are destined to system memory from either the processor or PCI. If the cache is programmed in write-back mode it will accept write data from the processor. The modified data then remains in the L2 until it is written to memory due to either an L2 replacement copyback or a PCI write snoop that hits in the L2. Updates to the L2 cache are done for processor burst reads from memory that miss, and processor to memory burst writes.

CENTRAL CONTROL

The central control unit performs the internal arbitration, coordinates the internal and external snooping, and controls the flow of transactions through the MPC105. The MPC105 uses internal buffering to store addresses and data moving through the part, and to maximize opportunities for concurrent operations. For most operations, the data is latched internally in one of seven data buffers. The exception is processor accesses to memory, in which the data is transferred directly on the shared data bus. There are eight address buffers which correspond to the seven data buffers and the current processor-memory access. The addresses for incoming transactions from either PCI or the processor are compared to the latched addresses for internal snooping purposes. See Figure 2 for the general buffer organization.

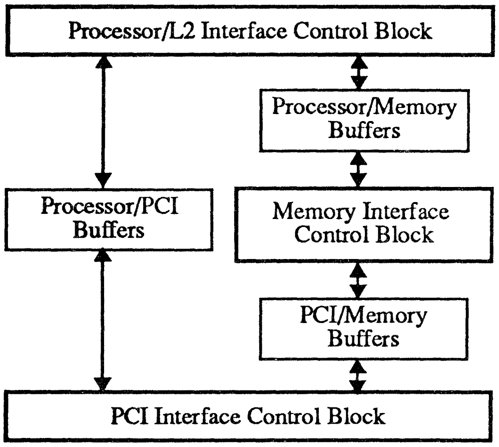

Processor/L2 Interface Control Block
Processor/Memory Buffers
Processor/PCI Buffers
Memory Interface Control Block
PCI/Memory Buffers
PCI Interface Control Block

Figure 2 **General Buffer Organization.**

Processor/L2/Memory Accesses

Because systems using the MPC105 have a shared data bus between the processor, the L2, and memory, for most cases it is unnecessary to buffer data transfers between these devices. However, there is a 32-byte castout buffer which is used for L2 castouts and for L1 copybacks due to snooping for PCI reads from memory. L2 castouts are caused when a processor transaction which misses in the L2, and the line in the L2 which will be replaced currently holds modified data. This data is latched internally to minimize the latency of the original processor-memory transaction. The slower flush of the data to memory is completed at the earliest available opportunity.

In the case of a snoop for a PCI read from memory which causes an L1 copyback, the copyback data is latched in the L2 copyback buffer and simultaneously forwarded to PCI. Once the L1 copyback is complete and PCI has finished reading from the L2 copyback buffer, the data is flushed to memory at the earliest available opportunity. Using the L2 copyback buffer for this purpose instead of the PCI read buffer simplifies the design by allowing fewer buffers to contain data which is modified with respect to memory.

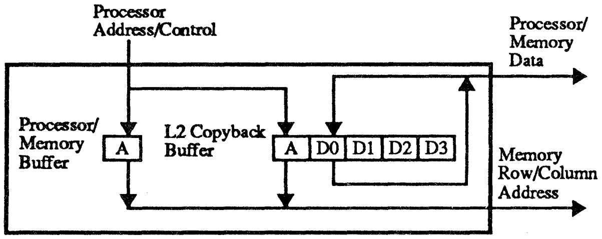


Figure 3 **Processor/Memory Buffers**

Processor Accesses to PCI

There are three buffers for processor accesses to PCI: one 32-byte buffer for reads from PCI, and two 16-byte buffers for writes to PCI. Buffering was required for processor reads from PCI for two primary reasons. First, the processor bus uses a critical-word-first protocol, while the PCI bus uses a zero-word-first protocol. The second design requirement was that the MPC105 be able to handle a memory access from an alternate PCI master if the target for a read disconnects part way through a data transfer. Because a PCI initiated read would require a snoop transaction on the processor bus, including a potential for a copyback, the processor address and data buses must remain accessible throughout the transfer of read data from PCI. Thus, all the requested data must be latched internally before the MPC105 responds to the processor. A buffer size of one cache line is required.

For example, if the processor initiates a critical-word-first burst read, starting with the second double-word of the cache line, the read on the PCI bus begins with the cache-line aligned address. If the PCI target disconnects after transferring the first half of the cache line, the MPC105 re-arbitrates for the PCI bus and, once granted the bus, will initiate a new transaction with the address of the third double-word of the line. If an alternate PCI master requests data from memory while the MPC105 is waiting for a bus grant, the MPC105 retries the processor transaction to allow the snoop for the PCI initiated transaction to be posted on the processor bus. When the processor snoop is complete, the subsequent processor transaction is compared to the latched address and attributes of the PCI read buffer to ensure that the processor is requesting the same data. Once all data requested by the processor is latched in the PCI read buffer, the data is transferred to the processor to complete the transaction.

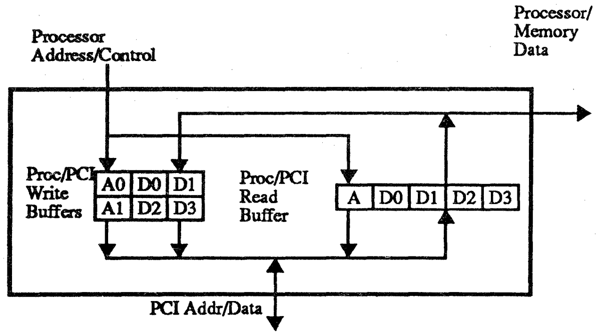


Figure 4 **Processor/PCI Buffers**

For processor writes to PCI, the primary design goal was to use internal buffering to minimize the affect of the slower PCI bus on the high speed processor bus. Once the processor write data is latched in internally, the processor bus can be available for subsequent transactions before the write is completed to the PCI target. Another design goal for these buffers was to effectively support both burst transactions and streams of single-beat transactions. The solution to these goals is a set of two 16-byte buffers, which can be used together as one 32-byte buffer for processor burst writes, or separately for single-beat writes.

In the case of a processor burst write to PCI, both buffers are used to store the processor data, and the address and transfer attributes are stored in the first address buffer. For a stream of single-beat writes, the data for the first transaction is stored in the first buffer and the transaction is started on the PCI bus. The second single-beat write is then stored in the second buffer. For subsequent single-beat writes, store gathering is possible if the incoming write is to the same half cache line as the previously latched data. Store gathering is only used for PCI memory address space, not PCI I/0 space, and can continue until the buffer is scheduled to be flushed or until the processor issues a synchronizing transaction.

PCI Accesses to Memory

All PCI accesses to memory are snooped on the processor bus to ensure hardware-enforced coherency between PCI, main memory, and the primary and secondary caches. For PCI reads, the primary design goal was to minimize the initial read latency, especially the effect of snooping on the read latency. Thus, when a PCI master requests data from memory, the memory access is started along with the snoop. If the snoop results in a hit in either the L1 or L2, the memory transaction is cancelled. PCI read data from the L2 or from memory is latched in a 32-byte PCI read buffer. For PCI reads which hit in the L2, the L2 sources the data without changing its internal state and no copyback to memory is necessary. For PCI reads which do not hit in either the L1 or L2, the data is fetched from memory starting with the requested address and continuing to the end of the cache line. Data is forwarded to PCI as soon as it is received, not when the complete cache line has been written into the PCI read buffer. New PCI read addresses are compared to the existing address, so if the new access is to the same cache line and the requested data is latched, the data can be forwarded to PCI without a snoop or another memory transaction.

To further minimize the latency for large block transfers, the MPC105 includes a selectable speculative read feature. When this feature is enabled, the MPC105 starts the snoop of the next sequential cache line address when the current PCI read is accessing the third double-word of the cache line. Once the speculative snoop response is known and PCI has completed the read, the data at the speculative address is fetched from memory and loaded in the buffer in anticipation of the next PCI request. If a different address is requested, the speculative operation is halted and any data latched in the PCI read buffer is invalidated.

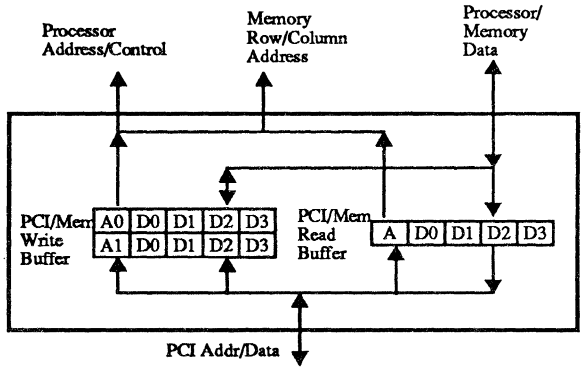


Figure 5 **PCI/Memory Buffers**

For PCI writes to memory the MPC105 supports two buffers that are each 32-bytes wide. For PCI writes to memory, the data can be latched internally without waiting for a snoop response. Thus write data can be accepted without any wait states. There are two buffers so that while a PCI master is writing to one, the other can be flushing its data to memory. Both buffers are capable of gathering for writes to the same cache line. If a snoop hit occurs, the copyback data is merged into these buffers into all the bytes that were not written by the PCI master. For write-invalidate transactions, a different snoop is used on the processor bus which causes the caches to invalidate any modified data without doing the copyback. Once the PCI write is complete and the snooping is resolved, the data is flushed to memory at the first available opportunity.

Arbitration.

There are two types of arbitration involving the MPC105, one for the PCI bus and the other for the shared processor/memory data bus. For the PCI bus, the arbitration is done externally and all processor-PCI transactions are performed strictly in-order. For the processor/memory data bus, the priority is as follows: processor memory read, processor/L2 transfer, L2 copyback due to a read snoop, PCI read from memory, processor memory write, snoop copyback due to write snoop, processor read or write from PCI, load of L2 copyback buffer, flush of internal buffer, speculative read. Note that if there is an address collision with an internal buffer, or if a buffer is needed but full, the flush for that buffer is bumped up in priority accordingly.

OTHER FEATURES

Power Management

For power management techniques to be successful all levels of the system, both hardware and software, must be involved. The MPC105 does its part by being a fully static design (typical power dissipation about 1.0 Watt) and providing four different power saving modes. It supports three different levels of power management through software programmability (Doze, Nap, and Sleep) and a suspend mode which is enabled through an input pin. The four power levels differ in the number of functional units remaining enabled as well as the type of transactions that are responded to by the MPC105. In Doze mode, all functional units are disabled except for PCI address decoding, system RAM refreshing, processor bus monitoring, and interrupt monitoring. Doze mode can be entered independently of all other hardware in the system. Nap mode is used when the system would like to reduce power consumption in both the processor and the MPC105. Sleep mode is also used in conjunction with the processors power saving modes and allows for the shutdown of the clocking logic within the MPC105 to further reduce system power consumption.

Error Handling and Test Logic

The MPC105 provides error detection and reporting on the three primary interfaces (processor interface, memory interface, and PCI interface). Errors detected by the MPC105 are conditionally reported (through programmable configuration bits) to the processor through the assertion of a machine check or a transfer error acknowledge. The system error and parity error signals are used to report errors on and to the PCI bus. The type of errors detected are: illegal transfer types from the processor, illegal flash ROM write transactions, memory parity errors, accesses to memory addresses out of the range of physical memory, PCI address and data parity errors, PCI no device select errors, and PCI received target abort errors. The address and type of transaction which caused the error are latched and held within the MPC105 so that diagnostic software can access them.

The MPC105 supports the IEEE 1149.1 JTAG standard providing a pin boundary-scan capability in a board test environment. Additional logic throughout the design allows for 99% test coverage in Motorola's manufacturing environment.

CHIP AND PACKAGING TECHNOLOGIES

The MPC105 is implemented in a 3.3 volt Motorola CMOS process with four levels of metal and a minimum drawn feature width of 0.65 µm. The MPC105 integrates over 256,000 devices on a 5.8 X 6.7-mm die size and is packaged in a 304 pin Ceramic Ball Grid Array (CBGA) that is bonded using Motorola's Control Collapse Chip Connection (C4) technology. This package reduces the parasitic package inductance by over 60% as compared to standard wire bond quad flat pack (QFP) packages. The package measures 21 X 25-mm and contains a solder ball array laid out to a pitch of 1.27mm (center to center). This technology significantly reduces the footprint area, one fourth the size of a QFP with a similar I/O count.

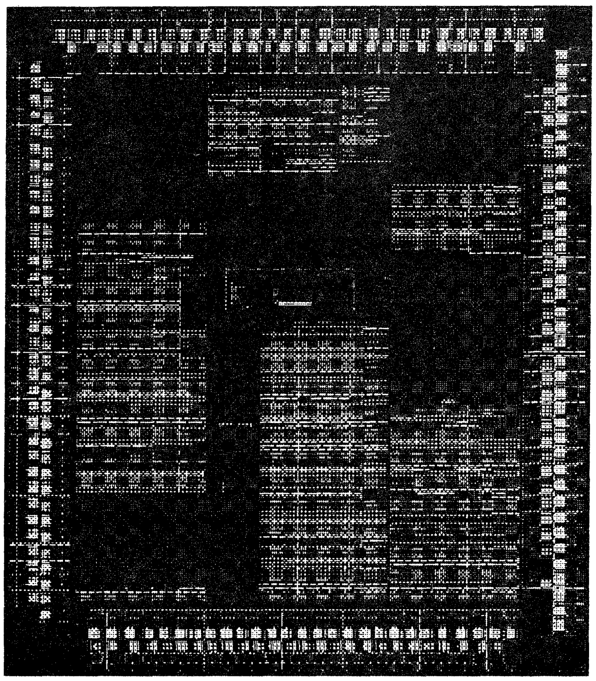


Figure 6 **MPC105 die photo.**

CONCLUSION

THE MPC105 is a fully integrated, high performance, PCI compliant single-chip host bridge and memory controller for the PowerPC 60x family of microprocessors. The flexible design is capable of performing in a wide range of systems from portables and handhelds, to workstations and multiprocessing systems. The flexibility of the memory and processor interfaces allows a designer to choose the memory system and processor which is most suitable for a system's performance needs and provides an easy migration path for system upgrades.

ACKNOWLEDGMENTS

The authors would like to acknowledge the hard work and dedication put forth by the design and verification engineers who worked on the MPC105 project. We would also like to thank the packaging team, the tools group, the manufacturing team, and all the other support staff.

REFERENCES

[1] Garcia, M.J. and Reynolds, B.K., "Single Chip PCI Bridge And Memory Controller For PowerPC Microprocessors," *Proc. of the 1994 IEEE Int'l Conf. Computer Design (Cambridge, MA, October 10-12, 1994), pp. 409-412.*

[2] MPC105 PCI Bridge/Memory Controller User's Manual, Motorola Incorporated, 1995.

[3] PCI Special Interest Group, "PCI Local Bus Specification *Revision 2.0", April 30, 1993.*

* PowerPC is a trademark of International Business Machines Corporation.

The authors may be reached at chrisb%ibmoto.com@oakhill.sps.mot.com,
garcia%ibmoto.com@oakhill.sps.mot.com, lauraw%ibmoto.com@oakhill.sps.mot.com

<u>Author biographical statements</u>

Christopher Bryant works for Motorola at the Somerset design facility in Austin, Texas, where he has been involved in the conception and definition of the MPC105 and also with the logic design of the PCI interface He holds a BS degree in Microelectronic Engineering from Rochester Institute of Technology.

Michael J. Garcia works for Motorola's RISC Division at the Somerset design facility in Austin, Texas, where he was a technical design manager for the MPC105 development team. He is currently working as the project leader for a future PowerPC product. Garcia holds a BSEE and an MSEE degree from Purdue University. He is a member of the IEEE and the IEEE Computer Society.

Laura Weber works for Motorola at the Somerset design facility in Austin, Texas, where she worked on the logic design for the central control unit of the MPC105. She holds a BS degree in electrical engineering with the computer option from the Ohio State University. She is a member of the IEEE and the IEEE Computer Society.

PCI TEST BENCHES FOR FPGA IMPLEMENTATIONS

John Birkner
Chairman, Steering Committee
PREP Corp.
Dir. CAE Products
QuickLogic Corp.
2933 Bunker Hill Lane
Santa Clara, CA 95052.
phone 1-408-9872022 email john@qlogic.com

Abstract

PLD suppliers are challenged to meet the needs of systems designers who are using programmable logic in PCI applications. Designers want to know:

- Are PLDs electrically compliant with PCI bus-drive specs?
- Can PLDs meet PCI speed and density requirements?
- Can PLD suppliers provide HDL reference design examples?
- Can synthesis provide designs meeting PCI timing requirements?
- Must critical paths be hand crafted?

HDL simulation Test Benches can answer many of these questions. Specifically, the Test Bench reports functional and timing parameter simulation results for a given post-place-route netlist and timing file. The designer can modify/optimize his HDL code and use the Test Bench to verify the quality of the design.

Besides discussing these key questions, this session will explain and discuss the concept of using Test Benches to evaluate design tools, programmable devices and specific implementations.

PREP Corp. is developing a set of Test Benches to use in comparing the efficiencies of different synthesis techniques, using many types of functions including the PCI bus interface.

This discussion will therefore also be a milestone report on the progress of that work.

PREP is a non-profit consortium of vendors of programmable logic devices and related design tools. PREP has developed standard benchmarks for comparing the performance and functional capacity of programmable logic devices. It's primary technical activity is now focused upon the Test Bench developments.

ARCHITECTURAL ENHANCEMENTS OF NEXT GENERATION PENTIUM™ CHIP SETS

by

John M. Monti
Director of Marketing
Symphony Laboratories
4000 Burton Drive
Santa Clara, CA 95054

ABSTRACT

The next generation of core logic chip sets for PC motherboards supporting Pentium-class microprocessors will add a host of new functional enhancements in order to increase overall system performance. These enhancements generally fall into two categories: improving memory performance as well as the I/O capabilities and bandwidth of the chip set. The caching mechanisms of future systems will be enhanced via support for larger cacheable main memory, SRAM bank interleaving, and direct interfacing to new synchronous burst SRAMs. Main memory performance will be improved through the implementation of new high-volume memory architectures such as Extended Data Out (EDO) and burst EDO DRAMs in addition to increased levels and sophistication of read pre-fetching and write posting. Input/output performance and capabilities will be improved via increasing the bandwidth available on the Peripheral Component Interconnect (PCI) Local Bus to greater than 100 Mbytes/second, adding enhanced peripheral controller cores, such as dual-channel bus master Integrated Drive Electronics (IDE) on PCI, and increasing the number of PCI bus masters and slots achievable in a system.

CACHE MEMORY

There are commonly two types of cache memory in higher performance PC systems: level one (L1) cache which is internal to the microprocessor, and level two (L2) cache which is external to the CPU. Since all Pentium-class CPUs contain from 8 to 16 Kbytes of L1 writeback or writethrough cache, the ability to support these is intrinsic to all of today's Pentium-class core logic chip sets. However, there are several enhancements which can be implemented in the second level cache controller built into the core logic which will further improve the system performance of next generation systems.

Cache Size

First of all, the chip set needs to support the most common cache sizes desired by end users, which is 256 or 512 Kbytes today on Pentium-class motherboards, in addition to adding support for future upgrades. As a result, the chip set should have the capability to directly access at least 1024 Kbytes, if not 2048 Kbytes for those chip sets targeted at the high end or server market. The amount of cacheable main memory should increase in accordance with the larger cache sizes up to a maximum of 256 Mbytes with a 1024 Kbyte L2 cache. Unfortunately, commodity 20 or 25 nanosecond asynchronous cache SRAMs, while very cost effective, may not provide the required performance with a 66 MHz external microprocessor bus, necessitating wait states. One way to reduce the wait states typically required by back-to-back cache accesses to different banks is to implement a bank interleaving scheme with the L2 cache, similar to the way DRAM bank interleaving had been implemented in the past. While improving the situation a small amount, dramatic improvement is possible only via use of a pipelined burst SRAM architecture.

Pipelined Burst SRAM

For this reason, Intel and the manufacturers of other high performance CPUs have recently advocated the manufacture of so-called synchronous burst SRAMs with the goal of keeping up with the ever-increasing speed of Pentium and future microprocessors. Beginning in 1994, non-pipelined burst SRAMs became available from a limited number of SRAM suppliers offering very high access speeds, such as 12 or 15 nanoseconds. Unfortunately, the manufacturing yields of these devices was below the percentages required to compete on a cost basis with the standard asynchronous SRAMs. As a result, this architecture has remained in small niche applications. A second movement, toward pipelined burst SRAMs, was precipitated by Intel later in 1994. The pipelined burst SRAM architecture allows a much slower SRAM array access by synchronously pipelining data and comfortably meets the requirements for a 66 Mhz 3.3V Pentium processor bus with today's standard CMOS process. This should improve yields substantially on the mainstream 3.3 Volt process technologies used to build today's SRAM devices in volume.

Product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

As a result of the improved manufacturing efficiencies of the pipelined variety, support for this type of SRAM will become a mandatory feature of Pentium-class core logic chip sets during 1995.

MAIN MEMORY

Main memory performance is fundamental to achieving higher overall system performance. In many mainstream systems which do not use any external cache memory in order to reduce costs, the impact main memory has on the overall system performance is magnified. In a typical cache-based Pentium-class system, 256 or 512 Kbytes of level two cache SRAM is typically used, adding an additional $20 to $50 to the hardware cost of the system. Although the majority of business PCs sold today still contain external cache memory, a decreasing percentage of consumer-targeted systems are including this costly addition. Primarily for this reason, DRAM vendors have been proposing a wide assortment of new memory products to maintain system performance while reducing total system cost.

Additionally, the main memory architecture can improve the average access time in systems with L2 cache during cache miss cycles. Various unique DRAM architectures are currently being proposed by the leading DRAM manufacturers as well as a handful of start-up companies. Most of these architectures will not be designed into high volume systems due to the increase in costs associated with them. This category of new architectures includes new synchronous DRAMs in addition to products from Rambus, MoSys, Samsung, and others. Several of these architectures are nonetheless expected to become pervasive in applications requiring very high bandwidth, such as local bus graphics and video boards, but the movement of these new memory architectures to main memory will be slow during the next two years.

Extended Data Out

The key to the future success of a new memory architecture in a main memory application is its ability to significantly improve the overall performance of the memory subsystem with little or no impact on system cost. For these reasons, it is expected that new Extended Data Out (or 'EDO') DRAMs will become mainstream products in the next generation of high volume personal computers and thus must be supported by the next generation of Pentium-class core logic chip sets. The key to EDO's improved performance is the elimination of an overlapped precharge time penalty between back-to-back sequential memory accesses. Although EDO DRAMs require the same amount of clock cycles as standard Fast Page Mode (FPM) DRAMs on the first access, the access time of each subsequent read or write is greatly reduced allowing zero wait state performance at a clock speed of up to 50 MHz. In a 64-bit memory system, transfers at this speed can peak at close to 500 Mbyte/second bandwidth during an access.

The manufacturing cost of EDO DRAMs is very close to that of standard FPM devices since the circuitry required to implement the overlapped precharge access is a small modification to the design and complexity of FPM. This allows the DRAM vendor to embed both EDO and FPM circuitry on the same die with little or no die size increase, as well as, use the same lead frame, plastic package, test hardware, marking equipment, shipment packaging, etc. This almost eliminates any additional production costs of EDO over FPM devices.

Burst EDO

Burst EDO DRAMs take the EDO concept one step further by embedding a processor-compatible, four-cycle burst count on chip supporting interleaved or linear bursts. This way the cycle time of sequential rows is reduced even more, resulting in 20 - 40% higher performance. With these types of DRAMs, zero wait state accesses at 66 Mhz are possible. Another advantage of burst EDO products is their ability to easily co-exist in the same system with standard EDO and FPMs. This capability must be supported by next generation core logic chip sets in order to ensure easy upgradeability, allowing the end user to upgrade his system without worrying about any incompatibility problems.

Read Pre-fetch & Posted Writes

Other architectural enhancements which will improve 64-bit memory system performance can be made in the chip set in addition to supporting new types of DRAMs. Two of the most beneficial are the ability to access data from memory before it is actually needed, called pre-fetching, and holding data in posted write buffers to be written later when the long DRAM first-access time can be hidden by other CPU activity. Due to the demanding needs of 90 and 100 MHz CPUs today, read pre-fetching and posted write buffers should be four or more accesses deep. In a 64-bit memory subsystem this means four quad word (36 bytes including parity bits) deep buffering.

PCI BUS BANDWIDTH

Most veteran PC users know that the promised performance improvement of the Peripheral Component Interconnect (PCI) local bus has been less than remarkable. In fact many graphics cards tally better benchmark numbers on the VESA local bus than PCI. The reason for this is not the definition of PCI or the design practices of system OEMs. The true culprits are the core logic chip set suppliers.

In a rush to provide PCI capability on 486 systems, several core logic vendors introduced so-called 'VL-to-PCI bridge' solutions into the marketplace. These chip sets enabled the manufacture of VIP motherboards which contain a slot for everyone: VESA, ISA and PCI. The problem is that hastily designed bridge chips add considerable latency to accesses across them. This extra latency can be obvious when high bandwidth peripheral cards are plugged into the PCI bus, such as new multimedia video boards. Additionally, most bridge chip solutions are not able to perform burst accesses across the PCI bus, especially not continuous burst accesses nullifying one of PCI's main advantages over VESA in multimedia applications requiring uninterrupted video data streams.

Advent of Deep PCI Core Logic

With the movement of Pentium-class microprocessors into mainstream computing applications, the requirement to support the aging VESA bus has been diminishing. For this reason, next generation core logic needs to support a 'deep PCI' architecture, which removes the additional bridge latency of first generation solutions. These chip sets must support zero wait state data transfers from back-to-back multiple bus masters on the PCI bus. There are several architectural enhancements which can be implemented in chip sets to increase the PCI bandwidth available closer to the theoretical maximum. For any PCI system, this limit is somewhere above 120 Mbytes/second on a 33 MHz, 32-bit PCI implementation.

Concurrent Operation

Concurrent operation makes possible several types of data movement within the same system during the same period of time. For example, a bus master controller on PCI containing its own on-chip DMA controller, should be allowed to directly transfer data to a target slave device on the PCI bus at the same time that the system CPU is performing other non-PCI tasks, such as reading or writing main memory. A SCSI controller may write directly to a LAN controller across PCI for instance, as part of an automated network back up utility. This transfer is allowed because architecturally, PCI is not a 'true' local bus since both the CPU and its local memory are on the far side of a processor bridge device from the PCI master/slave pair. However, in practice, many core logic chip sets cannot support this type of operation since a complicated system simulation must be performed during the design of the chip set state machine. This type of simulation is only possible using a high level design tool such as Verilog, which is not commonly used by many chip set manufacturers.

A second type of concurrent operation is the ability of the chip set to allow the CPU to update its local cache memory at the same time a PCI bus master controller is accessing a different region of the system's main memory. Of course, a complex algorithm must also be employed in this case in order to ensure the validity of data being accessed at all points within the system at the same time. For example, the chip set cannot allow a PCI bus master to read information from a location in main memory, if that same location, as represented in the L1 or L2 cache, was updated on the previous cycle. The chip set must contain a sophisticated method of continuously snooping all of the data located in the caches as well as any pre-fetch or posted write data buffers within the chip set to determine on a cycle-by-cycle basis which data is 'valid' or 'dirty'.

Concurrent operation improves overall system performance as well as specifically improving the bandwidth available on the PCI local bus. It is really a form of multi-processing, which has the potential of improving system efficiency in the same manner that the first 8237 DMA controllers had done so for adapter cards on the legendary ISA bus. In a concurrent system, the core logic must treat all data buffers throughout the various data paths in the system as small caches rather than mere data FIFOs. Concurrent operation is increased by placing buffers between the CPU and main memory, the CPU and PCI, as well as between PCI and main memory.

BUS MASTER IDE

In tomorrow's multimedia systems, which require continuous video streams to pass across the PCI local bus using tremendous bandwidth, mass storage performance can be improved by integrating a bus master PCI-IDE controller into the core logic rather than the more common slave-only type. Intel's Architecture Labs division has actively promoted the use of a standard architecture bus master PCI-IDE solution as a way to reduce one of the most common performance bottlenecks in today's Pentium-based systems. Bus master DMA added to the IDE function has two main benefits to overall system performance: reducing the number of interrupts to the CPU during large block data transfers, and the amount of PCI bus bandwidth required to transfer the data. In addition, the implementation of two separate IDE channels directly on the PCI bus enables multi-threaded I/O processes.

Reduced Interrupts

The original ATA standard required the disk drive to assert an interrupt after each 512 Byte sector is transferred. As a result, just the data transfer portion of a 1M Byte data move required a minimum of 2000 interrupts. Each of these had to be cleared by the system CPU by reading the IDE controller's status register and vectoring to an interrupt service routine prior to continuing the transfer. The time it takes to clear the interrupt can often be adversely affected by the operating system, the graphical user interface shell (i.e. Windows), as well as the application running on top of it. More recently, the ATA standard has evolved to allow larger block transfers which can reduce the number of interrupts somewhat. However, the ultimate interrupt reduction comes from integrating a DMA controller into the IDE controller chip.

A bus master DMA device is able to transfer almost an unlimited amount of data without requiring intervention from the system CPU. Even 64 Kbyte page boundaries can be jumped by the scatter/gather DMA controller until the complete transfer is finished, provided no error condition has occurred. The DMA controller together with its device driver can actually 'scatter' logically sequential data over various pages located in different physical areas of memory. When this data is required by the CPU, the DMA controller and driver act together to retrieve and reassemble it in its proper form for modification or transfer. Since all of this activity is handled without generating interrupts to the CPU, the total is greatly reduced. As an example, the number of interrupts required to complete a transfer can be reduced from several thousand to 2 or 3, leaving the microprocessor to complete other tasks at the same time.

Burst Transfer Capability

The second main advantage of using a bus master DMA IDE controller on the PCI local bus is its ability to reduce the PCI bandwidth required to move large amounts of data. Since the bus master device and its device driver have the intelligence to store data in an on-chip DMA FIFO, the bus master will only request access to PCI when the FIFO is approaching its capacity. Once the PCI arbiter grants access to the IDE controller, which can in reality be several microseconds after the bus access was originally requested, the PCI-IDE controller's DMA engine can perform zero wait state burst data transfers to or from memory at 33 MHz. Since even the fastest IDE disk drives today can only maintain a head transfer rate of 4 or 5 Mbytes/second once the cache on the disk drive is empty, the amount of the PCI bandwidth used is very small (roughly 5/125 = 4%). This reduction of bandwidth usage will allow bandwidth-hogging multimedia applications to function more efficiently in the same system.

Integration of a bus Master IDE controller into the core logic chip set will become a requirement by the middle of 1995. Due to the relatively large software requirements for the device drivers needed to control this portion of the chip set, several chip set companies are developing discrete versions of the bus master IDE controller which can be fully debugged and supported by software prior to its integration into a future chip set.

Multi-threaded I/O

Placing two separate IDE channels onto the PCI bus can improve the overall system performance perceived by the user. If the core logic implements all of the control and data paths to each port separately and proper support is built into the driver software and operating system, then true multi-threaded I/O processes are possible, similar to SCSI. Two channels allow the faster hard drive to sit on the primary port while a much slower CD-ROM can sit on the secondary port. Multi-threading allows each device to execute a command simultaneously, such as a format of the hard disk and a copy command from the CD-ROM. Presently this capability is only available in lower volume PC operating systems such as Windows NT, Novell NetWare, SCO Unix, etc., but will move into the mainstream with the increased acceptance of Windows 95 and OS/2 Warp during 1995 and beyond.

PCI ARBITER

The actual number of PCI components and slots available in any given system is a characteristic of the core logic chip set as well as the maximum allowable electrical loading specification of PCI. The lowest cost place in the system to integrate the PCI arbiter is in the CPU bridge device. By maximizing the number of bus masters supported by the chip set, the flexibility for the system designer to place a varying number of PCI peripherals directly on the motherboard in an LPX 'all-in-one' form factor is increased. If a bus master IDE controller is located in the chip set, the arbiter should dedicate one REQ/GNT pair to the IDE controller. Additionally, the arbiter should dedicate a second REQ/GNT pair for any ISA bus masters which are placed into the system by the end user. The DMA channel used for the ISA masters should be sharable by more than one DMA capable ISA card in case more than one is attempting to move data at the same time.

Since the core logic chip set usually takes two PCI loads (one for the CPU bridge and one for the SIO/ISA bridge device), eight possible loads are left for additional components. This leaves a maximum of four slots if no other PCI components are on the motherboard. If two PCI components are on the motherboard, such as a bus master IDE controller and a graphics controller, a total of three slots can still be supported if the arbiter has a total of six REQ/GNT pairs. This is why next generation chip sets should have six pairs. It increases the options for the motherboard designer without overly burdening the chip set manufacturer with too many additional pins (2 - 4).

Programmability

In order to maximize the efficiency of the PCI arbiter, it should be able to take into account the intrinsic differences between the various PCI master components on the local bus. For instance, top priority on a networked business machine should be given to the ethernet controller card since ethernet is non-deterministic and dropped packets can greatly reduce the overall performance of the local area network. Even though the IDE controller may seem more 'important' to the efficient functioning of the PC, the BIOS writer must take into the account the specific application in which the PCI arbiter will be used.

If the core logic design enables flexibility in the way the PCI arbiter can be programmed, enhanced system (and network) efficiency can be obtained. This is why in addition to a default round-robin arbitration scheme, the arbiter defaults should be overridden in 'degrees' (i.e. highest priority gets the most local bus bandwidth, next highest priority gets the next most, etc.). In this way, in a saturated PCI application such as video conferencing over a local area network, a LAN controller could maintain 40% of the PCI bus bandwidth, with the video controller getting the next 30%, and every other peripheral sharing the rest.

3.3 AND 5 VOLT AUTO-DETECTION

System designers have many choices for including various types of DRAMs, cache SRAMs, PCI boards, and CPUs onto tomorrow's Pentium-class motherboards. There are both 3.3 Volt, as well as, 5 Volt alternatives in each of the above four categories dictating that core logic should support both of these voltage ranges in the future. In order to minimize the design impact of including this capability, the chip set should automatically detect the voltage level of each of the four different bus interfaces: CPU, DRAM, SRAM, and PCI. This is easily done in the chip set by including I/O cells which change the voltage level they can drive and receive depending upon the Vdd sensed at local power pins on the device. For instance, each bus interface buffer set should have its own dedicated Vdd pins. When the Vdd pins corresponding to the I/O buffer cells are powered to 3.3 Volts, then the I/O buffers automatically drive 3.3 Volt levels. The input buffer should still maintain 5 Volt tolerance to avoid unnecessary silicon damage if 5 Volts is inadvertently driven into the chip.

SUMMARY

Next generation core logic chip sets for Pentium-class PC motherboards will require enhanced capabilities and performance in order to optimize the available hardware for demanding multimedia system applications. These enhancements include the capability of directly interfacing with future mainstream memory architectures, such as 3.3 Volt pipelined burst SRAMs and burst EDO DRAMs, integrating dual channel bus master PCI-IDE, providing a programmable PCI arbiter which can control up to six PCI masters, and automatically detecting the voltage level on each of the four key system interfaces: CPU, L2 cache, main memory, and PCI local bus.

BIOGRAPHY

John M. Monti received a B.S.E.E. from Yale University in 1985. He later earned a Master of Science in Engineering Management from Santa Clara University in 1990. After starting his career as a field engineer for Lockheed Missiles & Space Co., he worked for Advanced Micro Devices, Inc. for the next seven years in various marketing management positions. John joined Symphony Laboratories, a core logic chip set supplier, in 1994 as the director of marketing. The company is located at 4000 Burton Drive, Santa Clara, CA, 95054 and can be reached at 408-986-1701.

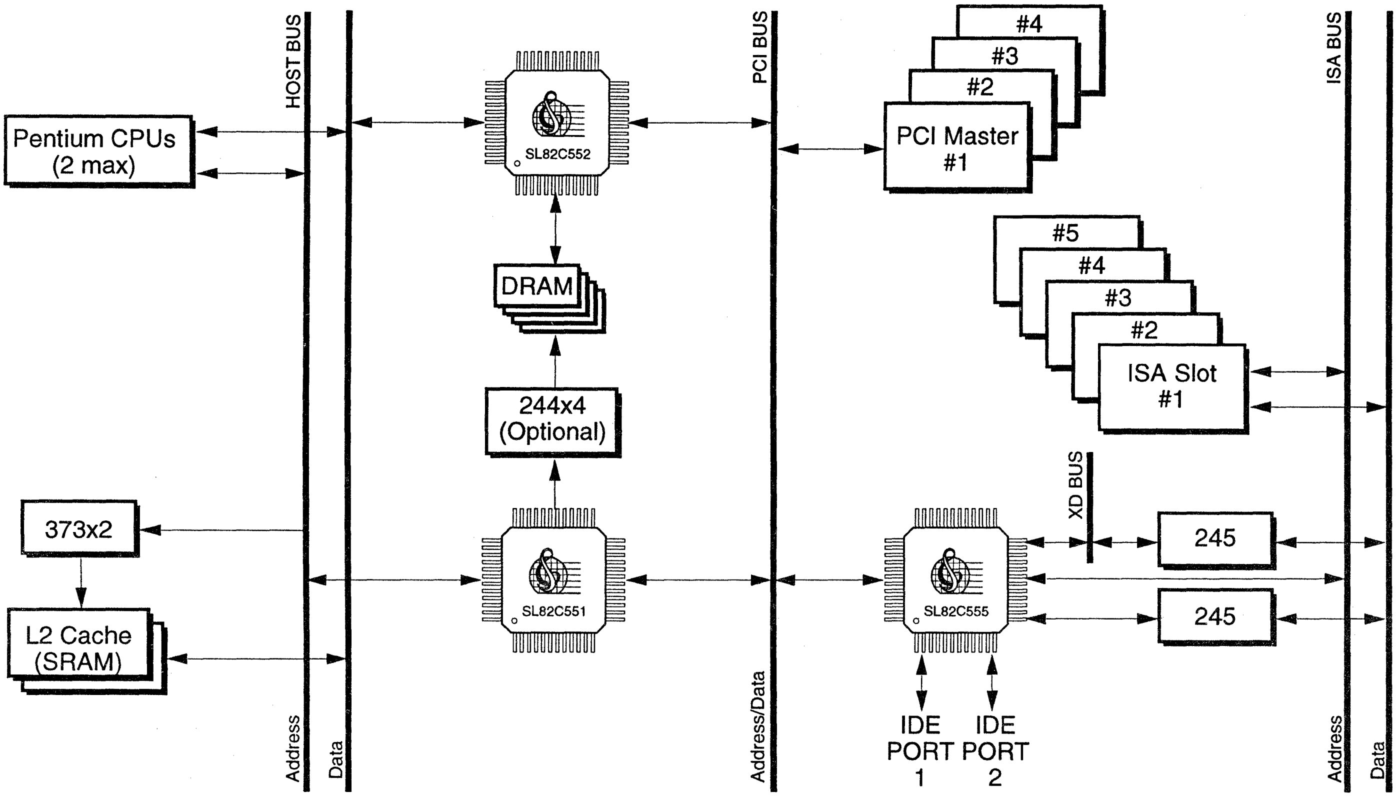


Figure 1. Symphony's Rossini Chip Set Solution

OPTIMIZING THE PCI/MEMORY INTERFACE FOR HIGH PERFORMANCE

James D. Joseph
Ramtron International Corporation
1850 Ramtron Drive
Colorado Springs, CO 80921

ABSTRACT

In a high-performance PCI system, the mismatch in data bus speeds creates the potential for a data transfer bottleneck. The problem is exacerbated with slow DRAM main memory - even if a fast secondary cache is included. Most system simulations only include the CPU/main memory interface. However, the interface between the local bus and main memory often contributes to a loss in overall performance of 10% or more.

The Ramtron Enhanced DRAM (EDRAM) memory with Quickcache™ is the ideal main memory component to simultaneously support 33 MHz PCI bus clock rates and even higher CPU data bus speeds. Its fast 15 ns page access time requires no wait states for any PCI read from main memory. The 15 ns write time allows a fast CPU to write to main memory directly without a write buffer. This performance can be achieved with a noninterleaved memory consisting of EDRAM SIMM modules. Fast SRAM memory normally used for secondary cache systems has similar performance but is 4 to 16 times lower in density and is many times more expensive. In addition, cache coherency issues are greatly simplified.

This paper illustrates a number of system issues involved in optimizing the various types of CPU/memory and PCI/memory cycles. In addition to memory timing examples for several processors, control strategies integrating an EDRAM controller with a PCI master, target, and host/PCI bridge are presented.

THE PROBLEM

CPU performance has increased dramatically in recent years. A major factor has been the increase in bus clock speeds. Unfortunately, DRAM main memory speeds have not kept pace. The result is a bottleneck - a fast CPU, capable of single-cycle burst reads and writes, must wait several cycles to read from or write to a 60- or 70-ns DRAM. System designers have alleviated the problem somewhat with SRAM secondary cache. As long as the CPU reuses data, the data can be read from secondary cache in one cycle. Although maintaining secondary cache coherency requires some overhead, simple CPU/memory or CPU/PCI target reads and writes are easily handled.

Another improvement is the writeback cache. This policy reduces memory write traffic on the CPU bus by writing to the L2 cache and/or DRAM only when necessary. The policy can be extended to the L2/DRAM and has no drawbacks in a simple system.

Unfortunately, the situation is not as simple in a system that includes PCI masters. Writeback is complicated by the snoop cycle. Cache coherency checking may not take any longer, but the valuable system memory is now not available for PCI master cycles. We want to optimize all six types of basic cycles as follows:

- CPU Reads from Main Memory
- CPU Writes to Main Memory
- PCI Master Reads from Main Memory
- PCI Master Writes to Main Memory
- CPU Reads from PCI Target
- CPU Writes to PCI Target

One comment - the 33 MHz, 5v implementation of the PCI bus is by far the most common. The discussion and timings presented here all assume use of that version.

ADDITIONAL EDRAM ARCHITECTURAL FEATURES

Before discussing each of the cycles in more detail, three other characteristics of the EDRAM architecture (see Figure 1) are useful for high-performance PCI designs. First, all reads come from the on-chip cache. For a read miss, the DRAM row accessed is loaded in parallel into the cache and the first word is accessed in 35 ns. All subsequent words in a CPU burst are read from the cache in 15 ns. With a read hit, the cache is directly accessed in 15 ns for the first and all subsequent words in a burst.

Next, the EDRAM allows cache reads in so-called "/CAL-high" mode. The column address is changed without latching and the data is read available 15 ns later. The 512Kx8 part has a selectable EDO feature to further increase the system designer's flexibility.

Finally, since during a CPU burst cycle all reads but the first must come from the cache, the DRAM array can be refreshed in the background, providing yet another way to improve performance.

CPU/MAIN MEMORY READS AND WRITES

Replacing conventional DRAM+secondary cache main memory with EDRAM allows bursts at SRAM speeds while substantially improving read miss performance. In addition, L2/DRAM cache coherency problems disappear along with the secondary cache.

Use of an EDRAM main memory also has important advantages for write cycles. Since DRAM writes are quite slow compared with the CPU bus clock cycle time, CPU support chipsets often include a four-deep write buffer to retire CPU-to-main memory writes with no wait states. This works well for processors like the Pentium and PowerPC; the only burst writes can only occur in writeback cycles, so the overhead of the writeback cycle means that the write buffer would never be filled. In other words, consecutive writeback cycles are impossible. For processors supporting more general burst writes (e.g., the MIPS R4600), the problem is more difficult; filling the write buffer is a definite possibility.

Use of a CPU-to-main memory write buffer with a PCI bus master also does not help if the PCI master wants to read from or write to DRAM; the master must wait until the DRAM completes the write. The faster EDRAM write cycles mean that a bus master has a much smaller potential memory conflict with an EDRAM main memory.

Finally, there is an important issue regarding writeback. In a system with ordinary DRAM, a writeback policy is almost always selected because DRAM writes are so slow it is best to minimize the total number of writes to main memory. With EDRAM and unbuffered zero-wait-state writes, the overhead associated with snoop cycles actually exceeds the EDRAM write transaction time, so writethrough may actually result in the fastest overall memory performance.

Table 1 summarizes the bus cycle counts for CPU/main memory reads and writes in a PowerPC 604 system with a 66 MHz CPU bus clock.

PCI MASTER/MAIN MEMORY READS AND WRITES

Because the PCI bus clock frequency is often lower than the CPU data bus frequency, the effect of PCI master/main memory wait states is magnified. However, zero-wait-state PCI reads, either single or burst, are not difficult even with conventional EDO DRAM technology, and timing is even more relaxed with an EDRAM main memory.

There are three issues with PCI master writes to a DRAM main memory. First, the EDO technology doesn't help with write cycles, so timing requirements are more critical. Second, since the number of words in a PCI burst write can be long, write buffering of master writes can't help to cover for a slow main memory. Finally, cache coherency is frequently an issue. Despite frequent chipset support for writeback protocol in the L2 cache, master writes are usually only to slower main memory. The next access to that data will always be a read miss. With an EDRAM main memory, not only is the memory capable of 15 ns consecutive writes within a page, but also coherency between the EDRAM's DRAM array and internal cache is automatically maintained.

CPU/PCI TARGET READS AND WRITES

CPU reads and writes involving a PCI target do not involve main memory, of course. Reads from the target provide little opportunity for creativity; the only option is to improve the target response. Two strategies are in common use. As with CPU-to-main memory writes, use of a write buffer clears the CPU bus more quickly. In addition, many of the memory controllers avoid continued PCI bus rearbitration by gathering numerous writes to consecutive locations into a single PCI burst write. Both of these functions should be retained in a system even if the main memory is upgraded to EDRAM.

Transaction	EDRAM (15 ns)	DRAM + SRAM cache
Burst Read Hit	4:1:1:1	6:1:1:1
Burst Read Miss	5:1:1:1	8:3:3:3
Write (Page Miss)	3-5 Cycles	3 Cycles
Write (Page Mode)	3 Cycles	3 Cycles

Table 1 - PowerPC 604 Bus Cycle Comparison @ 66 MHz Bus Clock

What about the memory controller design of a PCI system with EDRAM main memory? This is discussed in the next section.

EDRAM MEMORY CONTROL

Fast programmable logic can readily synthesize the memory control that takes advantage of the EDRAM's speed. We have used three approaches at Ramtron. First, signals from a built-in DRAM controller are "translated" by simple PAL devices (e.g., 22V10) to generate a new set of EDRAM control signals (Figure 2). This is most appropriate for low-performance applications (25 MHz CPU clock or lower). The next step is direct generation of multiplexed address signals by a PAL or fast PFGA (Figure 3). This approach works well in systems with 33-40 MHz CPUs. Ocean Information Systems uses this approach in the world's fastest PCI motherboard - an EDRAM-based DX4/100 system. For the highest performance, a combination of logic and fast FCT-E buffers assures no wasted cycles (Figure 4).

A good logic candidate for either or both functions is the Altera FX8160. In addition to a 6 ns tCO and a 10 ns tPD, the device provides extremely flexible clocking (three delay options and clocking off either edge).

Since the PCI bus clock is slower than the CPU clock, it is also critical to assure that cycles are not wasted while entering a PCI memory cycle. By integrating the logic for the EDRAM memory controller with the PCI master, target, or host bridge functions, we can further optimize performance. Intel provides FX8160 FPGA equations for PCI master or target logic in an application note (Brown and Heit 1994), and Ramtron extends this with FPGA equations for the host/PCI bridge logic and the EDRAM controller equations.

CONCLUSION

Replacing DRAM and secondary cache in a PCI system is a straightforward path to high performance. An EDRAM system achieves zero- or low-wait-state performance on all bus transactions. EDRAM outperforms the DRAM+secondary cache memory system while providing better density and lower memory cost with simpler writethrough cache policy. EDRAM provides the best cost/performance combination for PCI applications.

REFERENCES

Brown, C. and E. Heit, 1994, "Implementing PCI Interface Designs Using Intel's FLEXlogic FPGAs," Application Note AP-396, Intel Corporation, Folsom, CA (July).

PCI Special Interest Group, 1993. PCI Local Bus Specification, Revision 2.0, Hillsboro, OR (Apr.).

PCI Special Interest Group, 1993. PCI System Design Guide, Revision 1.0, Hillsboro, OR (Sept.).

Shanley, T. *PCI System Architecture*, Richardson, TX: Mindshare Press, 1994.

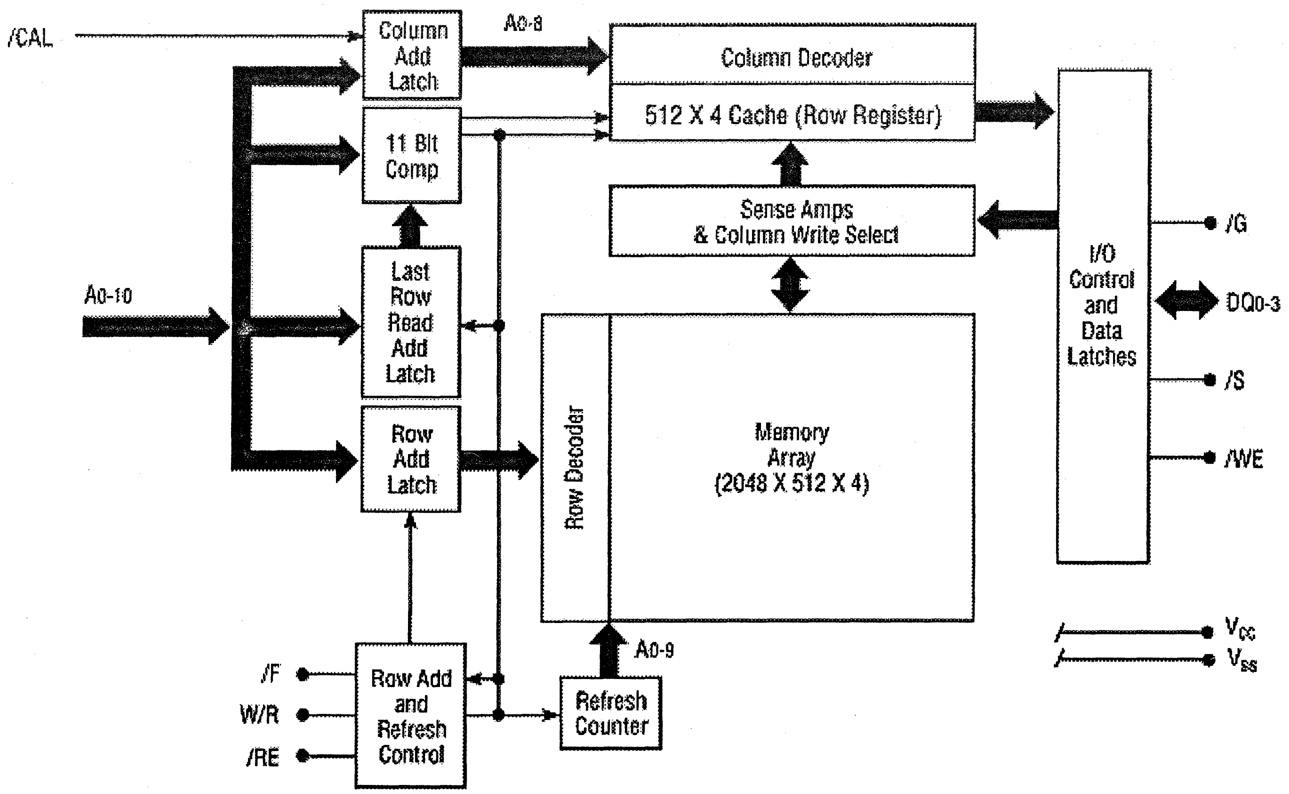


Figure 1
1Mx4 EDRAM Block Diagram

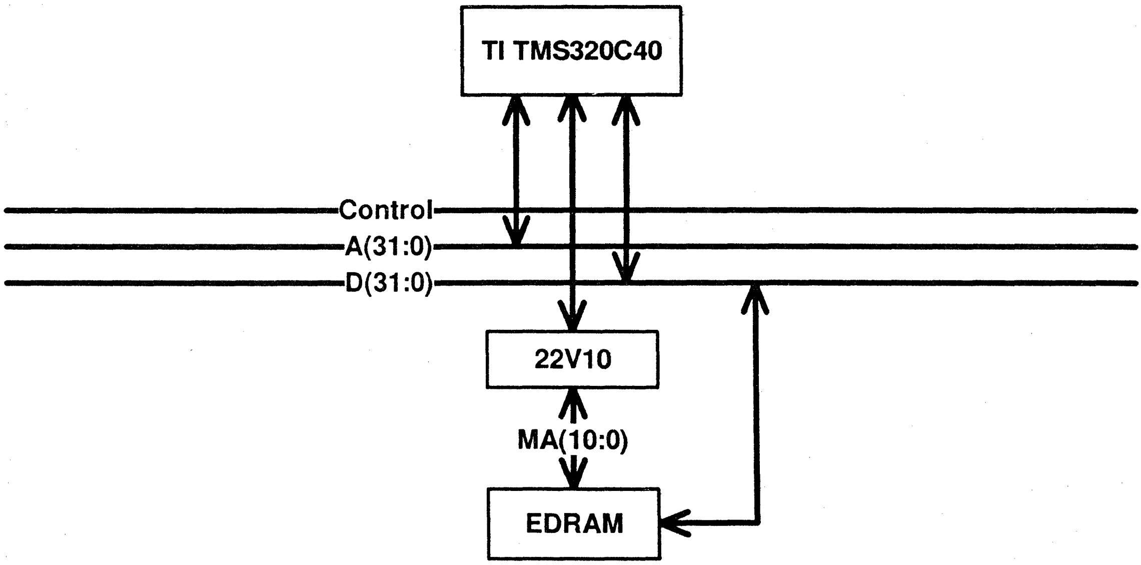


Figure 2
TMS320 System - EDRAM Control with 22V10

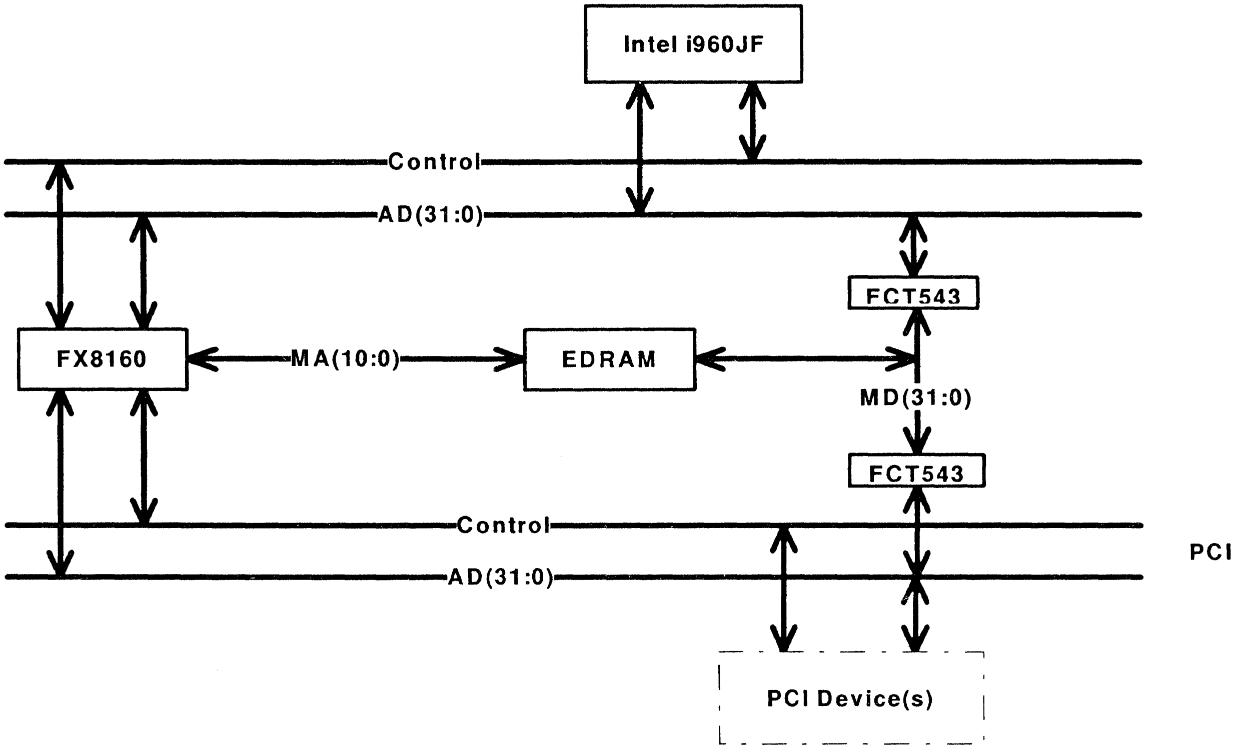


Figure 3
I960JF PCI System - EDRAM Control with FX8160

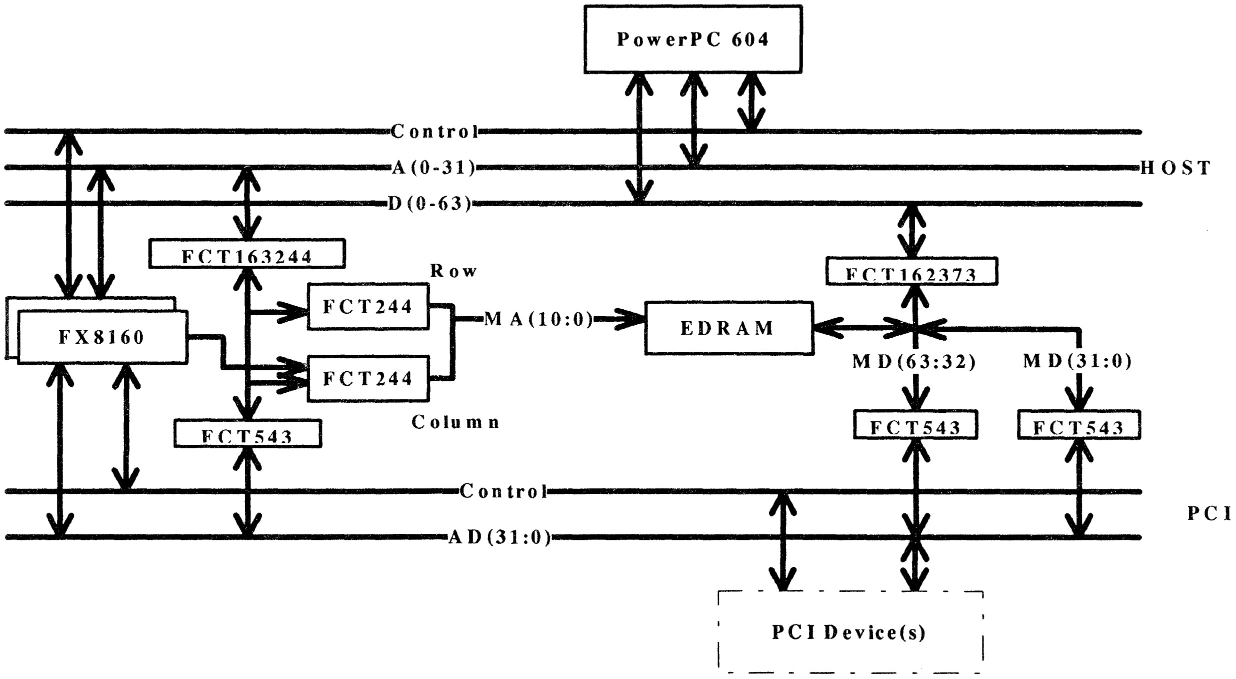


Figure 4
PowerPC 604 PCI System - EDRAM Control with FX8160 and Buffers

A PCI-Based Integrated Multimedia Chipset

David C. Baker, Ph.D.
Director, Texas Design Center
Brooktree Corporation
9868 Scranton Rd.
San Diego, CA 92121
(619) 452-7580/1249 (fax)

The BtV MediaStream family is a PCI-based integrated multimedia chipset that provides graphics acceleration, integrated wavetable synthesis audio, games compatible audio, live video in a window, video capture to disk, and full-screen, full-motion digital video playback - all on a single PCI load. BtV goes beyond the traditional graphics accelerator, achieving synchronization by accelerating all the multimedia data types in a way that is ideal for entertainment and game titles as well as for videoconferencing. The specific architectural considerations involved in the design will be discussed.

EXPANDING YOUR MARKET BY ADDING OPEN FIRMWARE SUPPORT TO YOUR PCI PERIPHERAL CARD DESIGNS

Greg Hill and Mitch Bradley
FirmWorks
480 San Antonio Road, Suite 230
Mountain View, CA 94040-1218
gregh@firmworks.com wmb@firmworks.com

ABSTRACT

Vendors of PCI add-in cards can expand their market beyond x86 machines by supporting Open Firmware.

This paper answers the following questions:

- What is Open Firmware?
- Why use Open Firmware?
  - How will it expand my markets?
  - How will it improve:
    - My development process?
    - My manufacturing process?
    - My field service process?
- How much engineering effort is required to create an Open Firmware driver?
  - What training is required?
  - What tools are required?
  - How long is a typical development cycle?
- What else must be added to my design for the PowerPC market?
  - For PowerPC Reference Platform machines?
  - For PowerMac?

DESCRIPTION OF OPEN FIRMWARE

Open Firmware is a portable boot firmware system. Boot firmware is the ROM-based software that controls a computer from the time that it is turned on until the primary operating system has taken control of the machine. The main function of boot firmware is to initialize the hardware and then to "boot" (load and execute) the primary operating system. Secondary functions include testing the hardware, managing hardware configuration information, and providing tools for debugging in case of faulty hardware or software.

Open Firmware, defined by IEEE Standard 1275-1994, is portable in the sense that its design is not tied to any particular processor family nor to any particular expansion bus. Open Firmware was specifically designed to support a variety of different processor instruction set architectures and different buses. Open Firmware is already in use on over a million machines, and is supported by several system vendors. A number of bus standards, including PCI, Futurebus+, VME-D, and SBus, contain provisions for Open Firmware card identification and booting. The PowerPC Reference Platform (PR*P) Specification requires that all PR*P-compliant machines shipped after June, 1995 use Open Firmware as their boot firmware (Dean and Adkins, 1994).

Firmware standardization can reduce system costs by eliminating "re-invention of wheels", providing "off the shelf" sources for firmware, eliminating unnecessary relearning of different firmware systems, reducing the effort of porting operating systems to different machines, and providing a consistent and powerful base set of hardware and software debugging tools.

The design of Open Firmware was undertaken as a long-term effort to "do it right", rather than viewing firmware as a "necessary evil" that should be done quickly and forgotten as soon as possible. As a consequence, it includes the following features.

Architected "Plug and Play"

Open Firmware's "plug and play" capability was designed in from the very beginning. Open Firmware provides auto-configuration capability more powerful than any previously available auto-configuration scheme, and is not tied to any specific vendor's products.

Open Firmware accomplishes this by providing support for self-identifying devices. Consider a computer with an "open" expansion bus such as PCI. An independent board vendor (i.e. not a system manufacturer) of a PCI card would like for the system to recognize and be able to use that card automatically. In the operating system environment, that may be easy; the board vendor can supply a driver on a diskette, and that driver may be loaded onto a hard disk or installed into the operating system.

In the firmware environment, acquiring drivers is more difficult; the firmware has to operate before the system is ready to read the disk. It is better to have the board driver in a ROM somewhere. Since a system ROM made today can't contain a driver for a plug-in card designed tomorrow, it is better to store such a driver in a ROM on the card with which it is to be used. This approach has been taken before, but in most existing firmware systems, the driver is stored as CPU-specific binary code, and thus only works on computer systems compatible with a particular CPU instruction set.

Open Firmware uses the "plug-in driver" technique, but instead of storing those drivers in machine language, Open Firmware encodes the drivers in a machine-independent language called "FCode". FCode is a byte-coded "intermediate language" for the Forth programming language. Forth, an ANSI standard interactive programming language, is based on a stack-oriented "virtual machine" that can be easily and efficiently implemented on any computer. FCode drivers are "incrementally compiled" into system RAM for later execution. The same FCode driver can be used on systems with different processor types. Thus, for example, a particular PCI add-in card could be used as a boot device in a PowerPC-based PCI system or in a SPARC-based PCI system with no firmware changes.

In addition to its use for firmware device drivers, FCode also provides a descriptive capability known as "properties". Plug-in cards use properties to report their characteristics to the firmware and system software. Such characteristics may include the device name, model, revision level, device type, register locations, interrupt levels, supported features, and any other identification information that makes sense for the particular device. The PCI bus binding even defines a property that can contain either the actual operating system drivers or the location of the OS driver, System software can use this property information to configure itself automatically for correct operation with a particular device. This information is stored in a processor/architecture-independent format that may be retrieved and decoded easily. Furthermore, this format is open and extensible, and allows any arbitrary device information to be recorded, providing protection against obsolescence of the interface.

Flexible Naming

Open Firmware was designed for adaptability. Its notation and structure for naming particular devices is based on a hierarchical "device tree" that mimics the bus configuration and physical addressing of the machine on which it is being used. This structure applies equally well to simple single-bus desktop machines and to "back room" servers with multiple processors and complicated hierarchies of interconnected buses. The "name space" for individual device names was designed so that no central authority is needed for "allocating" names - companies can design their products without appealing to a "master name arbiter".

The Open Firmware command language is open-ended. In addition to the standard commands that are present on all implementations, an arbitrary number of new commands may be added at any time, even by the user. Such additional commands might provide access to system-specific features, or might simply be customizations for the needs and tastes of individual users.

Configuration Maintenance

As mentioned, plug-in devices describe their own characteristics with FCode. Such descriptions are stored in the device tree. Each device tree node represents a particular device, and the description of that device is stored in its device node. Buses are considered to be devices in this sense, and are represented by "interior" nodes in the device tree. The "children" of a bus node represent the devices attached to that bus. Permanently-installed "built-in" devices also have device tree nodes with associated descriptions. The set of descriptive information about a particular device is open-ended, so new types of devices and new characteristics are added easily.

An operating system can use the device tree, with its device descriptions, to configure itself, locate particular devices, attach device drivers, etc. This supports the growing requirement for "plug and play" installation of new devices.

The Open Firmware "plug and play" capability is more comprehensive and less tied to specific buses, and operating systems than the Intel "Plug 'n Play" scheme, yet the Open Firmware scheme can co-exist and in many cases subsume the Intel scheme.

Interactive Interpreter

The core of Open Firmware is an ANS Forth-compliant kernel. This kernel provides the set of language primitives and operators used to implement the drivers and interfaces of the Open Firmware system. Compatibility with the ANS Forth Standard ensures source code portability and stability for future development.

The kernel contains an integrated machine language assembler/disassembler and a rich set of extensions for development and debugging at the hardware level, including breakpoints, stepping, tracing, disassembly, and memory and I/O operations.

Given these capabilities of Open Firmware, what are the practical benefits of adopting it?

MARKETING BENEFITS OF OPEN FIRMWARE

PR*P-compliant machines and the PCI bus Power Macintosh systems all provide PCI plug-in slots. All of these machines use Open Firmware. The size of the PR*P market remains to be seen, but Apple Computer has maintained approximately a 15% market share over a long period of time. With Apple converting to PCI as their standard bus, if a PCI card manufacturer who currently only targets x86 machines adds Open Firmware support, at least an additional 15% of the market opens up.

Not only does the market size expand, but customer satisfaction (which also affects one's market presence) is increased. Virtually everyone reading this paper is someone who has (probably more than once) had the unpleasant experience of trying to add a new peripheral card to an IBM-compatible PC. How many new purchases have been soured by a lost weekend spent trying to get things to work? How many of you gave up or settled for less than the promised performance because you just couldn't get it to work as advertised?

"Plug 'n Play" for the PC may help, but it's really just a partial solution that was bolted on 10 years after the fact, and which is nicknamed "plug and pray" for good reason. It wasn't designed in; it was added on. And, despite everyone's best efforts, it is a complex approach that doesn't quite succeed.

Open Firmware has architected "plug and play" that has been field-proven for over 6 years in over 1 million machines. Would your customers prefer products that were easier to install and configure? If your competitors provide Open Firmware support and you don't, will your customers stay with you?

ENGINEERING BENEFITS OF OPEN FIRMWARE

Eliminate Drivers for non-Boot Devices

The PCI Binding for Open Firmware provides for the system's Open Firmware to create Open Firmware properties automatically by reading the configuration space header of a PCI card. If the manufacturer fills out the header, non-boot devices need not contain Open Firmware drivers. Of course, non-boot devices may contain drivers, in which case that driver's properties can extend or override those automatically created from the configuration space header.

Fewer Boot Drivers to Write

In today's world, a PCI manufacturer of a boot device will have to write an x86 driver and an Open Firmware driver. That's the bad news. The good news is that as additional processor families using Open Firmware adopt PCI, no additional firmware work will be required to support those machines.

Write Boot Drivers Faster

FCode drivers do not execute in a vacuum: they may take advantage of Open Firmware operations supplied by the system firmware. The Open Firmware "device interface" specifies the full set of FCode primitives guaranteed to be available to an FCode driver. This set effectively constitutes an Application Binary Interface (ABI) for FCode which is required to be consistent and dependable across platforms and processors, providing a powerful framework for writing machine-independent drivers. Open Firmware system ROMs contain built-in support for frame buffers, network devices and protocols, disks, tapes and terminal emulation. By using these facilities, drivers are smaller, and are easier to write and debug.

The device interface also specifies probe and configuration practices on a bus-by-bus basis, affording processor independence to third-party developers.

Faster Debugging

Open Firmware's Forth kernel provides a wonderfully interactive environment for firmware development and testing. As a high level language, Forth is well suited to top-down design. Additionally, Forth's interpreter provides an environment in which bottom-up testing is easily accomplished, even very early in the development cycle. If you have ever struggled to understand a poorly written (or perhaps misleading, or even just plain wrong!) data sheet, you will appreciate the ability to write a routine and immediately try it on the hardware. And, if the routine fails, re-write it in a matter of seconds and try it again. Forth even provides a high-level-language "patch" capability to further enhance the ability to make rapid program changes in a development environment.

Not only does Open Firmware's interactive interpreter ease the life of the firmware developer, but the interpreter provides an environment in which hardware engineers can quickly bring up new designs by writing simple debugging routines "on the fly". Since Forth enables a designer to write programs quickly right on the test platform (without having to go through an edit-compile-link cycle elsewhere), data can be gathered quickly and new experiments can be run rapidly.

Open Firmware can debug hardware, operating system software, plug-in drivers, and even the firmware itself. The emphasis is on interactive tools for exploring problems, rather than "canned" diagnostics (although Open Firmware includes provisions for "canned" diagnostics as well). With today's short product cycles, a new design may spend as much time in the lab as in actual production. Open Firmware is an excellent bringup tool, and has been shown to shorten the time it takes to get a product to market.

MANUFACTURING BENEFITS OF OPEN FIRMWARE

Many companies that use Forth in the lab have found that the same diagnostics used by the developers for bring-up are well-suited for use in manufacturing, particularly during the initial ramp-up and/or for low volume products.

FIELD SERVICE BENEFITS OF OPEN FIRMWARE

Maintainability

Field ROM upgrades are expensive. Open Firmware provides a "self-patching" facility that allows many types of firmware bugs to be fixed without changing the system and/or device driver ROMs. The same facility allows additional firmware capabilities to be added to systems or drivers in the field, without changing the ROMs. (This capability might be viewed as less important given the increasing use of flash ROMs as the storage medium for firmware. However, when the overhead costs of a release and testing cycle are taken into account, Open Firmware's "one-off patch" capability can still be quite valuable.)

The Forth interpreter also provides a set of programmable debugging features to allow users and service personnel (as well as developers and manufacturing personnel) to isolate system problems in the event of a failure.

Configurability

Another service issue is storage and maintenance of user choices, such as the preferred boot device and the amount of memory to test. Open Firmware has "configuration variables" which keep such user choices in non-volatile memory, such as battery-backed RAM, electrically erasable PROM, or flash memory. Open Firmware configuration management uses self-describing, human-readable parameter names and values. The human readable values are encoded for efficient storage in the non-volatile memory device, where space is often at a premium. All access to these parameters is by name; new parameters may be added and old ones deleted at will, allowing for easy evolution of product families.

User Friendliness

Many system vendors are providing graphical front ends to make simple firmware interactions easy, but the full power of the Open Firmware command-based "user interface" is also present for technical users who need access to the full power of the system.

Nearly all the Forth kernel primitives are accessible interactively. The user interface provides access to the system for booting and debugging, and includes features such as command-line editing, ANSI terminal emulation, and system security controls.

ADDITIONAL DEVELOPMENT COSTS

Third Party Driver Development

For those companies without internal development resources with which to produce Open Firmware drivers, one solution is to hire a company like FirmWorks to create the required drivers on a turn-key basis. Even those companies that want to "grow their own" Open Firmware development capability can benefit from the use of third party drivers on their first Open Firmware designs. Having an experienced programmer create the first Open Firmware driver provides a specialized example from which inexperienced programmers can learn.

Training

Training in the writing of Open Firmware drivers can take many forms. The book *Writing FCode Programs for PCI* (available exclusively from FirmWorks) provides information on the specifics of FCode for PCI, including sample drivers. The book assumes some familiarity with the Forth language.

For those who want a classroom approach, one week classes in Forth and Open Firmware are available from FirmWorks. Classes can be held in Mountain View, CA or on your site. These classes assume no previous Forth experience. By the end of the week, participants will have written a simple driver.

For those who want a bit more help, a FirmWorks engineer can "look over the shoulder" of a customer's engineer while s/he writes an FCode driver.

Tools

Any PCI system running Open Firmware can serve as a development platform, provided that an FCode tokenizer is also available. One solution is an Open Firmware developer's kit for PCI from FirmWorks. Apple Computer also has suitable machines available to participants in their Power Macintosh developers program. When PR*P-compliant machines become available, they will also be suitable platforms when combined with an FCode tokenizer. Such tokenizers are available from FirmWorks and in Apple's Power Macintosh developer's kit.

Development Cycle Length

Development cycles vary with the complexity of the driver. Drivers for non-boot devices that simply publish property information can be completed in a fraction of a day. Relatively complex bootable device drivers can be completed in two weeks by an experienced programmer or in 4 - 6 weeks by a novice programmer.

OTHER POWERPC REQUIREMENTS

The most significant additional requirement is an OS driver. Windows-NT, AIX, Workplace OS and Solaris are currently expected to be available for PR*P machines. For the Power Macintosh, a MacOS driver is required.

REFERENCES

Dean, M. and A. Adkins, eds. *PowerPC Reference Platform Specification*, Version 1. Document number MPR-PPC-RPU-02. International Business Machines, 1994, p. 85.

Institute of Electrical and Electronics Engineers, 1994. Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices. IEEE Standard 1275-1994. IEEE, Piscataway, NJ

PCI Open Firmware Working Group, 1994. PCI Bus Binding to IEEE Standard 1275-1994. Available by anonymous ftp from playground.sun.com in /pub/p1275/bindings/postscript.

BIOGRAPHY

Greg Hill is director of marketing of FirmWorks, a consulting firm specializing in providing Open Firmware system ROMs, device drivers, and training. Prior to founding FirmWorks in February, 1994, Greg was a senior engineer with ROLM where he had been since 1974. Although his last assignment with ROLM was doing embedded firmware, he previously held positions at ROLM in digital, analog and ASIC hardware design, manufacturing test engineering and production engineering. Greg has a BSEE (with honors) and a BA (psychology) from Lafayette College and an MSEE from Stanford University.

Mitch Bradley is president and chief technical officer of FirmWorks. Prior to founding FirmWorks, he was the technical leader of the firmware group at Sun Microsystems Computer Corporation, where he conceived and developed the Open Firmware concept. He is chairman of the IEEE P1275 Open Firmware Working Group, which developed the Open Firmware standard, and vice chairman of the ANS X3/J14 Technical Committee, which developed the ANSI Forth standard. His educational background includes degrees in various technical disciplines from Vanderbilt University, Cambridge University, and Stanford University. Before embarking on the quest for Open Firmware, he did a variety of electronics things, from analog circuit design to operating system hacking.

EXPANDING SERVER I/O CAPABILITIES TO NEW PERFORMANCE LEVELS

Byron Gillespie
Intel Corporation
5000 W. Chandler Blvd. M/S CH5-233
Chandler, Arizona 85226

ABSTRACT

With the standardization of the PCI local bus, server designers are faced with many system-level design challenges. This paper focuses on improving server performance by moving the interrupt processing from the host processor to the I/O subsystem.

NEED FOR INTELLIGENT I/O INNOVATION

Today's client/server computing environment presents a challenge for maximizing server performance. Increasingly powerful host CPUs must be offloaded of interrupt processing in order to perform at optimal levels. Creating intelligent I/O subsystems based on high performance embedded processor allows the host CPU to perform more effectively.

Other factors driving the need for intelligent I/O subsystems include:

- The stand-alone computing model is being replaced by networked computing driving the need for network computing I/O.
- Networked computers increase the vast quantities of data the server systems support.
- Since the host processors (CPUs) in these servers also now run user applications, they need more powerful storage interfaces for accessing larger and larger disk storage areas in addition to higher reliability offered from RAID storage.
- Simultaneously, the data sizes and type increasingly contain natural data elements like video or audio, in addition to text and graphics.

Clearly, in today's client/server model, data congestion occurs more frequently at the servers. An intelligent I/O subsystem creates the balance between server performance and the data I/O paths and relieves data congestion.

EXAMPLES OF INTELLIGENT I/O INNOVATION

Storage I/O Interfaces

In the two basic areas of server I/O, network and storage, intelligent I/O subsystems offer superior server performance possibilities.

RAID controllers describe one of the better known examples of storage. The server can initiate a disk *store* or *retrieve* command as if it were writing to a single disk. The intelligent RAID controller separates commands into parallel *read* or *write* commands to its attached array of disks.

This parallel operation, controlled by the intelligent I/O processor, compensates for the single disk spin-up delay and protects data. This results in superior data transfer rates as well as greater reliability, a critical necessity as servers become more widely used for corporate computing and the creation of databases.

A similar example of how intelligent I/O improves server storage connection is in caching disk controllers. In this application, the host can write a data file to the intelligent disk controller cache at speeds matching the fast DRAM memory. The host application execution continues while the I/O processor controls the actual disk storage sequences.

i960® JF Microprocessor-based SCSI Disk Caching Controller

For example, an intelligent I/O disk caching controller can utilize a i960® JF microprocessor. This processor, rated at over 30 VAX MIPS, can implement advanced features like complex caching algorithms (intelligent read-ahead, dirty block invalidation), sophisticated disk management (scatter-gather, elevator sorting), and multithreaded I/O for multitasking operating systems.

Hardware Figure 1 shows a block diagram of a SCSI disk caching controller for the PCI Local Bus. The V961PBC[1] provides two independent DMA channels with bi-directional FIFOs supporting PCI burst transfers up to the maximum 132 Mbyte/sec rate. The V96SSC[2] companion chip to the i960® JF microprocessor integrates a (E)DRAM controller, four 32-bit counter/timers, 4-channel DMA controller, integrated interrupt controller, programmable chip selects, and debug serial port. For this application, the most important V96SSC features are the glueless interface to the memory and the DMA controller to transfer data between the DRAM and the SCSI controller.

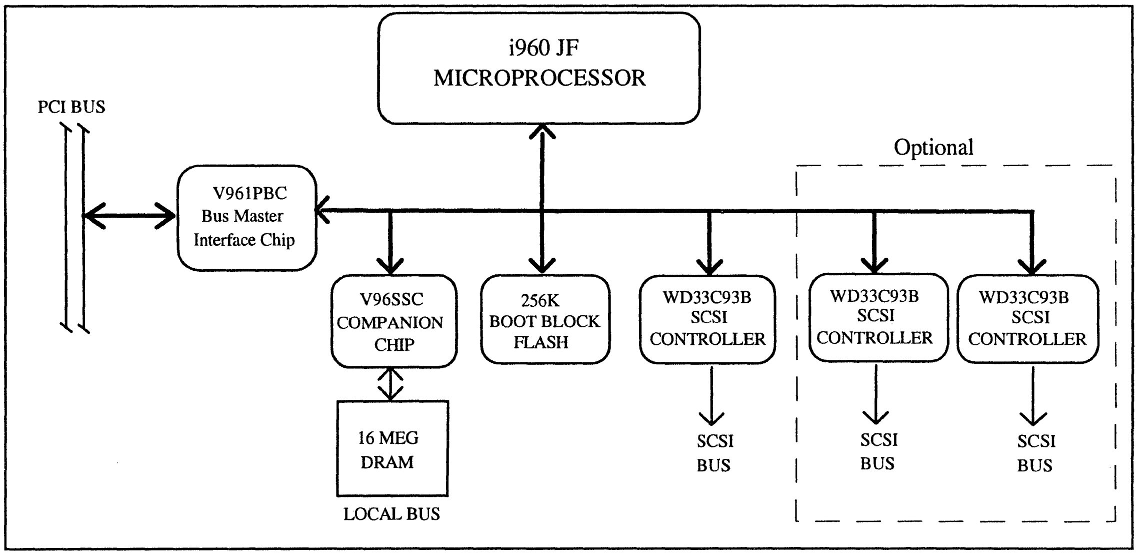


Figure 1. i960® JF Microprocessor-Based SCSI Controller

The V96SSC also contains a programmable I/O controller that generates control signals required to interface the i960 J series processors to common peripherals. All address range and access timing parameters are fully programmable. The V96SSC also supports the control logic for boot ROM.

The V961PBC has two independent bi-directional FIFOs allowing it to transfer data independently from the i960 JF microprocessor. The FIFOs loaded by one bus (local or system) empties onto the other bus when reaching the number of programmed entries in the FIFO or terminal count. The V961PBC DMA controller will release control of the local bus once its FIFOs are full/empty or when the local latency timer expires. The DMA controller will release control of the PCI bus when the FIFOs are full/empty or when the PCI latency timer expires and it loses the PCI grant signal.

The PCI host processor can directly access devices on the local bus for non-burst reads and writes. Configuration registers within the V961PBC will control the decoding and mapping of these accesses to the local address space. The i960 JF processors (shown in Figure 2) can also directly access the PCI bus. Again, configuration registers within the V961PBC will control the decoding and mapping of these accesses to the PCI bus address space.

Network I/O Interfaces

On the network I/O side, the bridge and routing functions are migrating into the servers. The Ethernet or token ring LAN interface with intelligent I/O handles the

[1] The V961PBC is from V3 Corp., Toronto, Canada

[2] The V96SSC is from V3 Corp., Toronto, Canada

frequent I/O interrupts and intelligently buffers messages to and from the host. This allows the host to streamline applications processing and to use other system resources, such as the system bus and memory, more effectively.

In WAN interfaces, intelligent I/O processing also offers significant performance improvements. For example, intelligent WAN interfaces can compress large message files prior to transmission. This not only frees the host CPU to perform more valuable application tasks, bus can allow more users to share the same fixed, wide-area connection. Also, the host CPU is transparent to this operation, thus providing better utilization of the WAN.

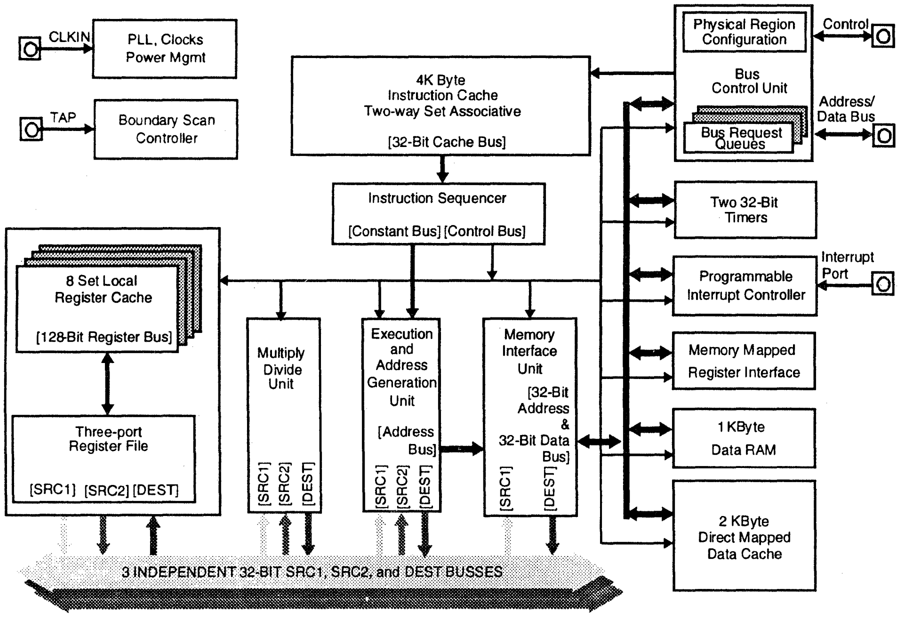


Figure 2. i960® JF Processor Block Diagram

i960® JF Microprocessor in Networking

There are several factors increasing the performance required for networking. ASCII data types heavily load today's networks. The LAN connection rate is growing twice as fast as the PC installed base. In addition, as the performance on the desktop increases, users expect quicker response. Finally, multimedia, with its high bandwidth and strict latency requirements, place huge strains on the existing network infrastructure. The i960 JF microprocessor delivers the performance needed to solve these networking problems.

In selecting a processor for a networking system, the developer requires a simple hardware interface and a robust tool chain to make porting existing code straightforward. Figure 3, shows a block diagram of a typical, medium performance, multi-port Ethernet bridge card. The boot ROM holds the code and may transfer it to DRAM for faster code execution. In addition, the SRAM typically stores interrupt handlers, protocol analyzers, and bridging/routing tables. The DRAM contains less critical code and data. The memory control section implements common system functions such as timers, real-time clock, interrupt multiplexing, DRAM control, and a system debug port.

The Ethernet component receives serial data from the physical interface and assembles the resulting packet. The on-chip DMA controller stores the data packet in memory. The Ethernet component can also transmit a packet from memory. One example of an Ethernet component includes the National Sonic.

The Ethernet interrupts the CPU after the first packet arrives. The CPU scans the received packet and checks the destination address against the bridging table in memory. The processor decides whether to forward the packet or ignore it. In addition, the processor analyzes the source address and updates the bridging table as needed (the learning function). After processing the packet, the system polls the other ports to see if another packet has arrived. The system allows bridging of packets between the two Ethernet chips and between other boards via the PCI bus connection.

The number of packets per second that can be bridged/routed is the critical performance benchmark for these systems. As multiple protocol routers become more intelligent, the need for higher system performance increases. The i960 JF microprocessor has several features that allow the networking developers to achieve the highest possible performance in a networking application. These include high bandwidth instruction and data caches, on-chip SRAM, multiple, independent execution units, branch prediction unit eliminates execution delays from the branch instruction, and parallel instruction execution.

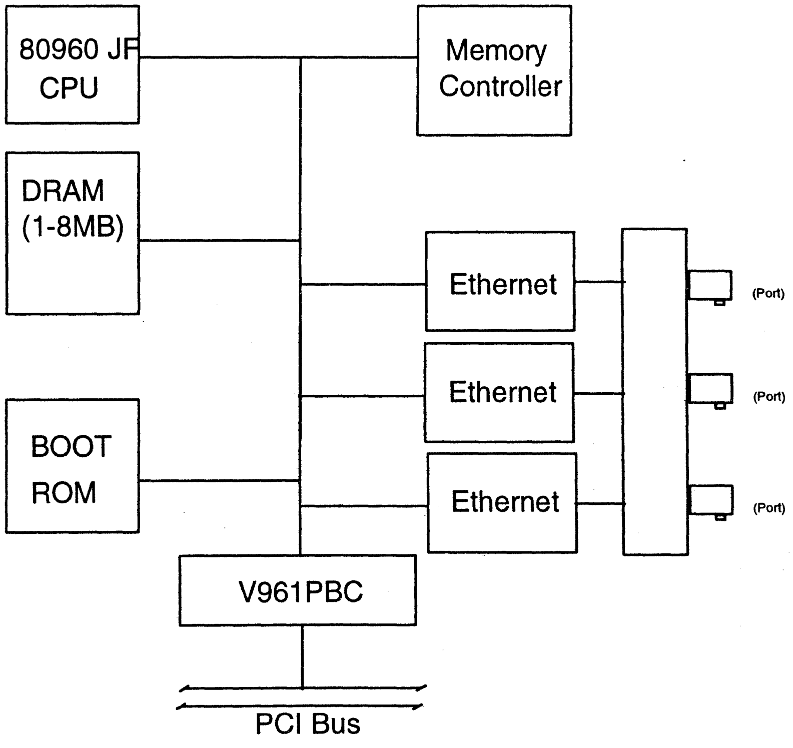


Figure 3. Simple Multi-Port Bridge

Big Endian Support The natural data type format for networking is big endian. This requires formatting the network data to conform to network protocol function and performance requirements. Incoming network data structure typically violates the traditional data alignment, or natural boundaries of the microprocessor. In such cases, the processor must perform "unaligned" accesses into big endian memory regions. The i960 JF processors support unaligned big endian bus requests. These include word accesses to a short or byte boundaries and short word accesses to a byte boundary.

Big endian byte ordering is supported in the i960 JF microprocessor, simplifying the development of applications that use big endian data. This enhancement allows the processor to access big endian data transparently without degrading the system performance. The user does not have to perform the byte swapping to reorder the bits. Unaligned big endian support eases the porting of code written for older big endian architectures, such as the 68000.

Full Featured Interrupt Controller and Interrupt Structure

Both the caching SCSI controller and Ethernet controller examples described require high performance interrupt processing to move the interrupt processing from the host CPU to the intelligent I/O subsystem.

The i960 architecture provides 32 executing priorities. The executing priorities decide the importance level of the executing tasks and interact with interrupt structure for redirection of program execution. An interrupt is an event that causes a temporary break in program execution so the processor can handle another task. Each interrupt vector has an associated priority. The requests for interrupt service come from many sources. Redirection of the executing task occurs if the interrupt request is higher priority than the executing task. These interrupt priorities allow the system designer the flexibility to implement an explicit structure to prioritize the various tasks associated with a high-performance embedded design.

The interrupt vector number accompanies the interrupt request. It indexes into the interrupt table to locate the entry point of the interrupt handler. From that entry, it gets an address to the first instruction of the selected interrupt procedure. The processor then makes an implicit call to that procedure. The processor switches to supervisor mode and changes stack to use a dedicated interrupt stack for the interrupt call. The processor allocates a new frame and a new set of local registers on the interrupt stack for the interrupt procedure. The processor saves the interrupted program's current state. Upon return from the interrupt procedure, the processor restores the interrupted program's state, switches back to the stack that the processor was using before the interrupt, and resumes program execution.

Every interrupt request contains an associated interrupt vector in the interrupt table. The table contains 248 vectors, from vector number 8, assigned the lowest priority, to vector number 255, the highest priority. The i960 architecture transparently prioritizes the 248 possible interrupts. There are 31 interrupt levels of priority, with eight vectors per priority.

The i960 J series processors integrate additional interrupt controller features. These processors contain eight interrupt input pins. These eight inputs support three different modes: dedicated interrupt mode with eight inputs; expanded mode that receives the interrupt vector from the eight input pins; and mixed mode which splits the eight inputs into three dedicated interrupt inputs and five expanded mode vector bits. The dedicated interrupt inputs can internally cache the interrupt vector entry in the on-chip data-RAM. In addition, these processors allow the application to cache the interrupt handling procedure internally in the instruction cache. Locking the interrupt handler in the cache and caching the interrupt vector provides the lowest interrupt latency for the dedicated mode interrupts. The i960 J series processor also contains a separate non-maskable interrupt .(NMI) pin. The NMI interrupt executes at priority 31, uninterruptable by any interrupt event.

All of these interrupt controller features with the hardware priority detection built-in the i960® J series processors result in fast interrupt response and flexibility for the system designers.

Interrupt Latency

The processor design provides interrupt processing latency and enough performance to eliminate the "software-based interrupt processing" typically performed by the host server CPUs. The RISC microprocessor in the hardware system relieves the system CPU of RAID chores and thus doubles the transfer rate as compared with the software system for the storage applications.

In networking I/O, the i960® microprocessor and service the networking controller interrupts without loss of packets. It also performs data buffering and compacting. This data buffering and compacting combines the data packets into larger, yet linear packets for transfer to the host system memory, thus effectively utilizing of the PCI bus.

COMPLETE TOOL SET FOR EMBEDDED DESIGN

There are many factors evaluated before choosing a RISC processor architecture. These considerations include processor performance, system costs, upgrade paths, customer support and a complete tools' package. The i960 architecture is supported by 200 development tools, services and support components.

SUMMARY

This paper has presented two typical intelligent I/O applications. From the cost-sensitive SCSI controller application to the performance-driven networking application, the i960 microprocessor family has the device to meet the need. The i960 architecture offers an easy to use, robust architecture, for the application designers to create powerful applications.

Additional Information

Applied Micro Circuits Corporation offers an PCI-to-80960 component. Contact AMCC 6195 Lusk Blvd. San Diego, CA 92121-2793

PLX Technology offers an PCI-to-80960 bus bridge component. Contact PLX Technology, 625 Clyde Ave. Mountain View, CA 94034

V^3 Corporation offers an PCI-to-80960 component. Contact V^3 Corporation, 2348 Walsh Ave. Suite G, Santa Clara, CA 95051

References

"i960® Jx series Microprocessor User Manual", Order Number: 272483-001 Intel Corporation, 1994.

"V961PBC Local Bus to PCI Bridge Controller Manual", V^3 Corporation, 2348 Walsh Ave. Suite G, Santa Clara, CA 95051

"V96SSC Local Bus to PCI Bridge Controller Manual", V^3 Corporation, 2348 Walsh Ave. Suite G, Santa Clara, CA 95051

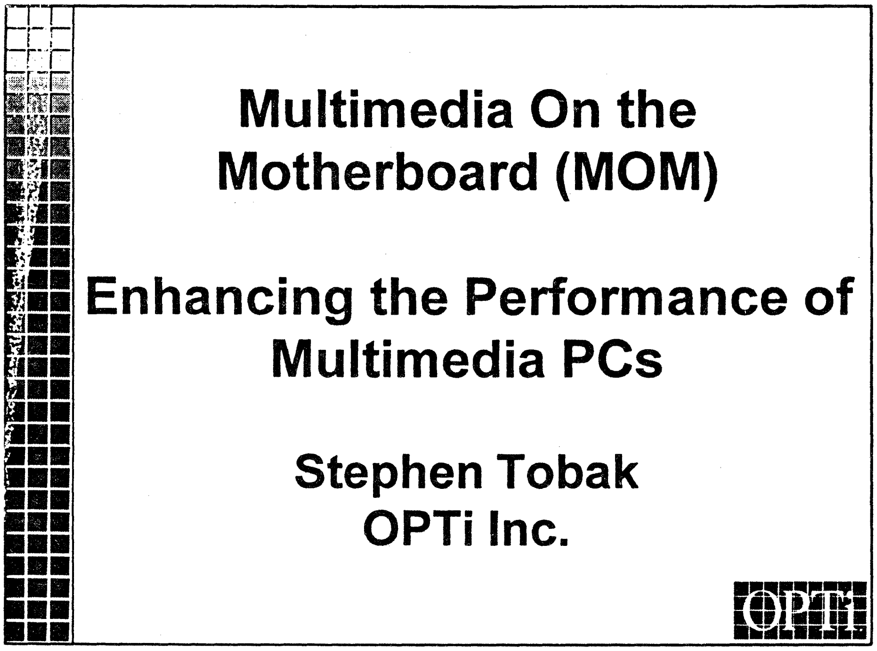
All other trademarks are the property of their respective owners.

BIOGRAPHY

Byron Gillespie works as a strategic development engineer for Intel's Embedded Processor Division. He is responsible for working with customers to define the requirements for future i960® microprocessor products. Before coming to Intel in 1991, he had eight years experience writing software for embedded avionics applications. The embedded applications used a variety of Intel processors including the superscalar i960® CA microprocessor. Gillespie received a B.S. in computer science from Northern Arizona University in 1983.

Multimedia On the Motherboard (MOM)

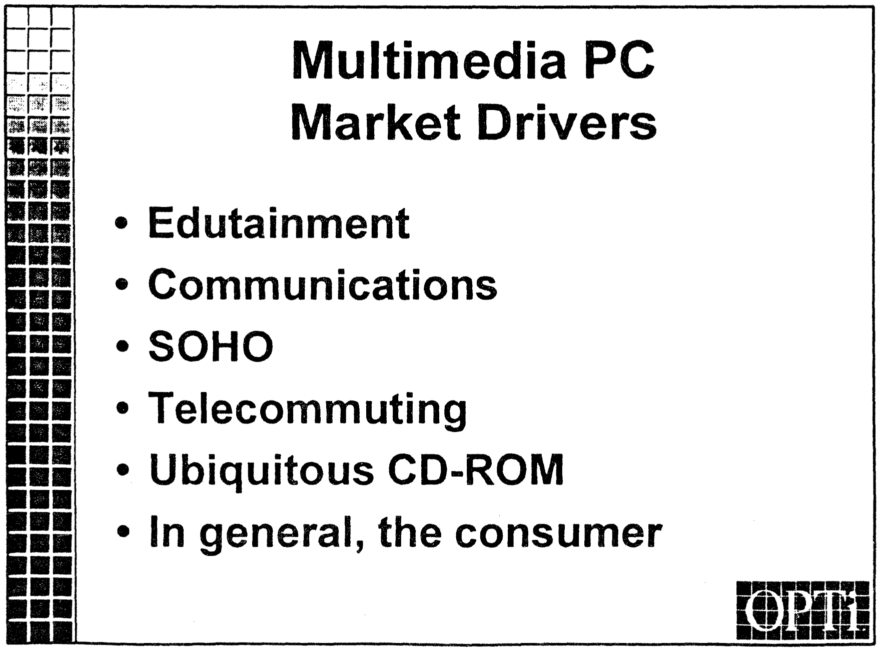
Stephen Tobak
Director of Corporate Marketing
OPTi Inc.
2525 Walsh Ave.
Santa Clara, CA 95051
408-486-8243



Multimedia On the Motherboard (MOM)

Enhancing the Performance of Multimedia PCs

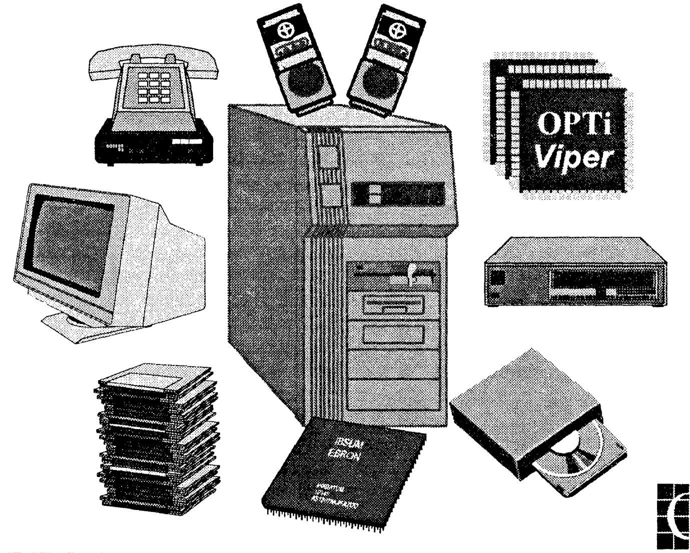
Stephen Tobak
OPTi Inc.



Multimedia PC Market Drivers

- Edutainment
- Communications
- SOHO
- Telecommuting
- Ubiquitous CD-ROM
- In general, the consumer

Multimedia System Components



Do Today's Systems Provide Adequate Performance

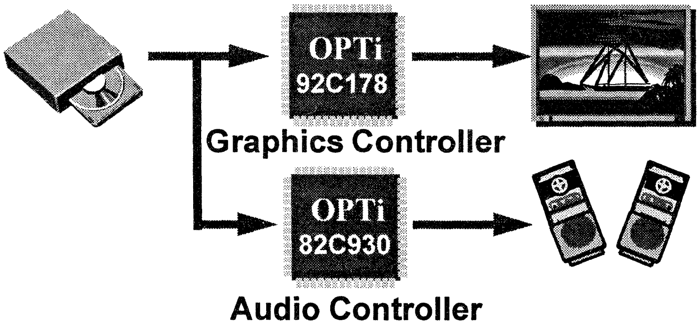
- CPU/Memory performance
- Graphics performance
- Benchmarks emphasize single function/task performance, keeping other functions constant
- Performance metrics do not adequately measure multimedia systems performance

System Bottlenecks

- Audio Playback from CD-ROM
  - ‘Lip Sync’ phenomena
  - Audio/Video Synchronization
- Video playback and overlay
  - Hardware assist required...
  - Without adding cost
- Multitasking OS’s
  - Shared system resources
  - CPU / Memory bus bandwidth

OPTi

Bottleneck Example: CPU / Memory Bus Bandwidth Audio / Video Sync



OPTi 92C178

Graphics Controller

OPTi 82C930

Audio Controller

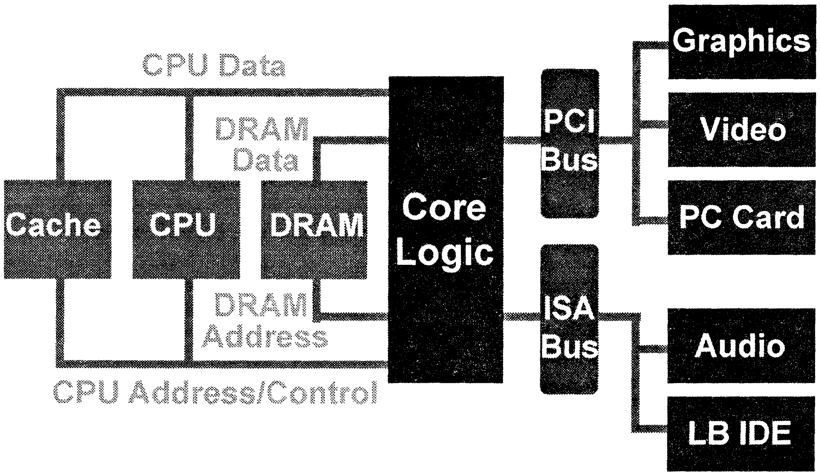
- Video/Graphics problem - granularity
- Audio problem - lip sync

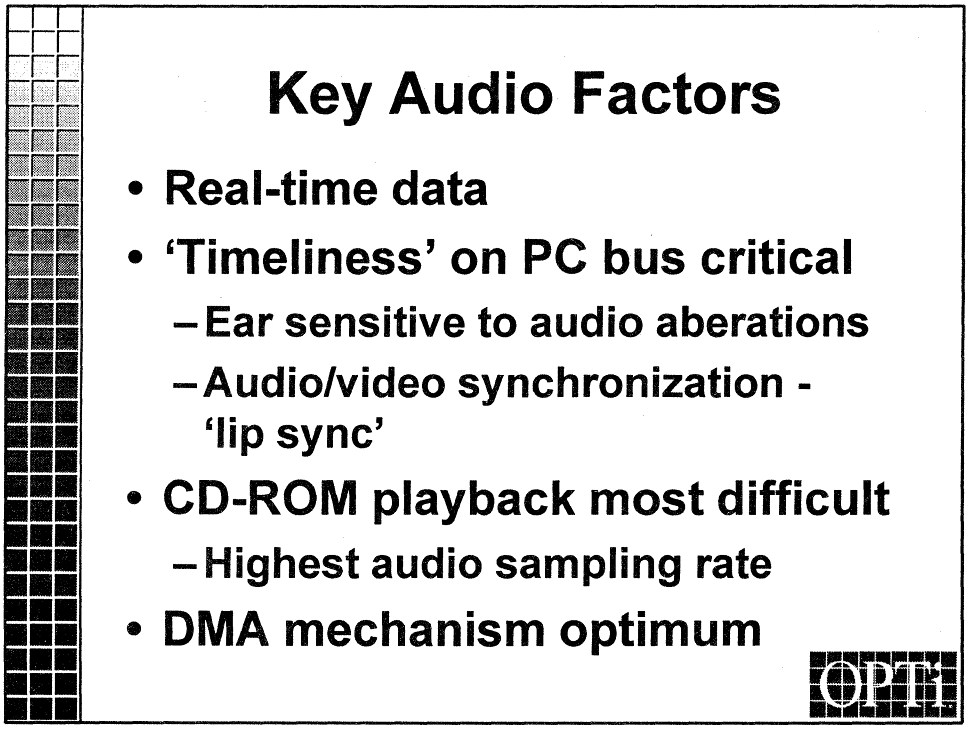
OPTi

The Goal

- To free up as much CPU / memory bandwidth as possible by improving multimedia interface throughput
- Enables CPU to perform other tasks, i.e. NSP
- Result is improved system performance on today's multimedia applications, with bandwidth left over for tomorrow's applications

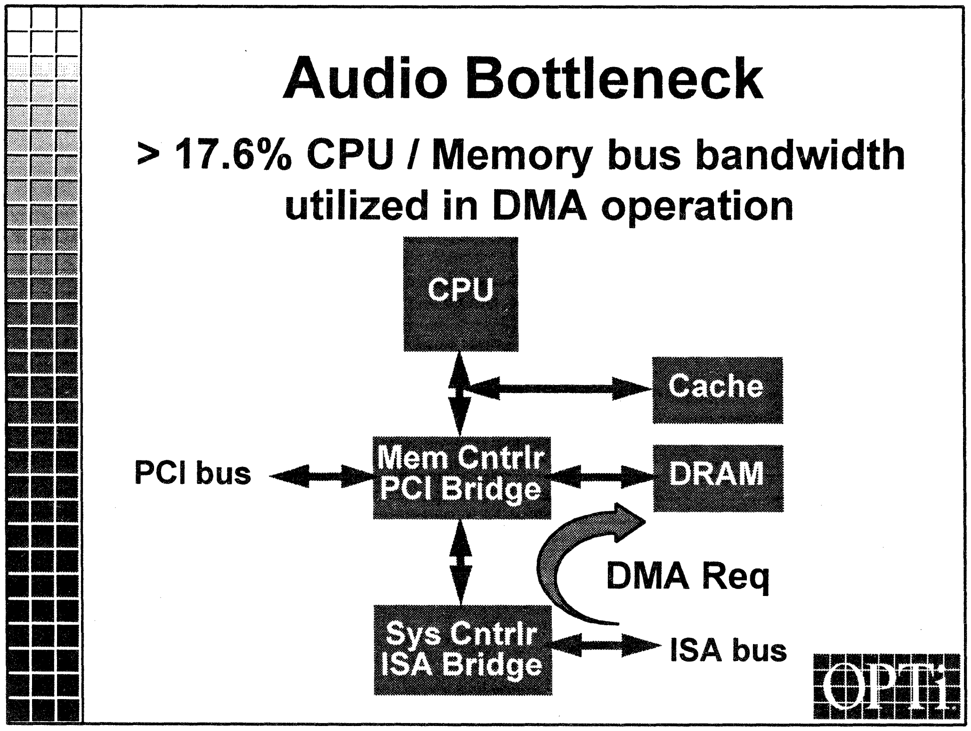
Multimedia System Architecture





Key Audio Factors

- Real-time data
- ‘Timeliness’ on PC bus critical
  - Ear sensitive to audio aberations
  - Audio/video synchronization - ‘lip sync’
- CD-ROM playback most difficult
  - Highest audio sampling rate
- DMA mechanism optimum



Audio Bottleneck

> 17.6% CPU / Memory bus bandwidth utilized in DMA operation

CPU

Cache

PCI bus

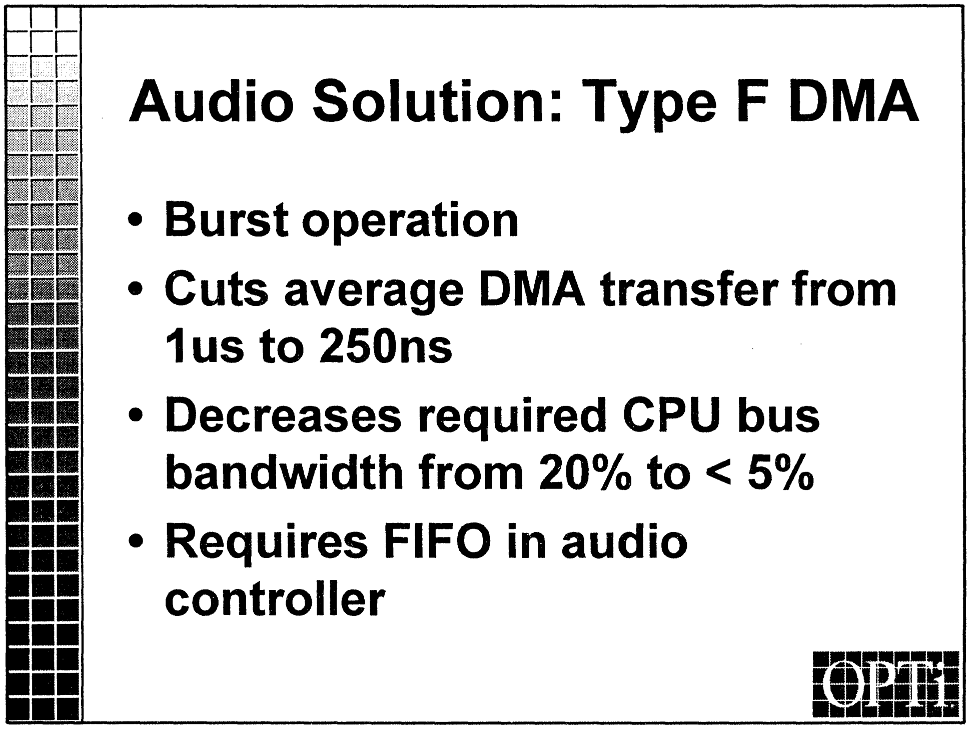
Mem Cntrlr
PCI Bridge

DRAM

DMA Req

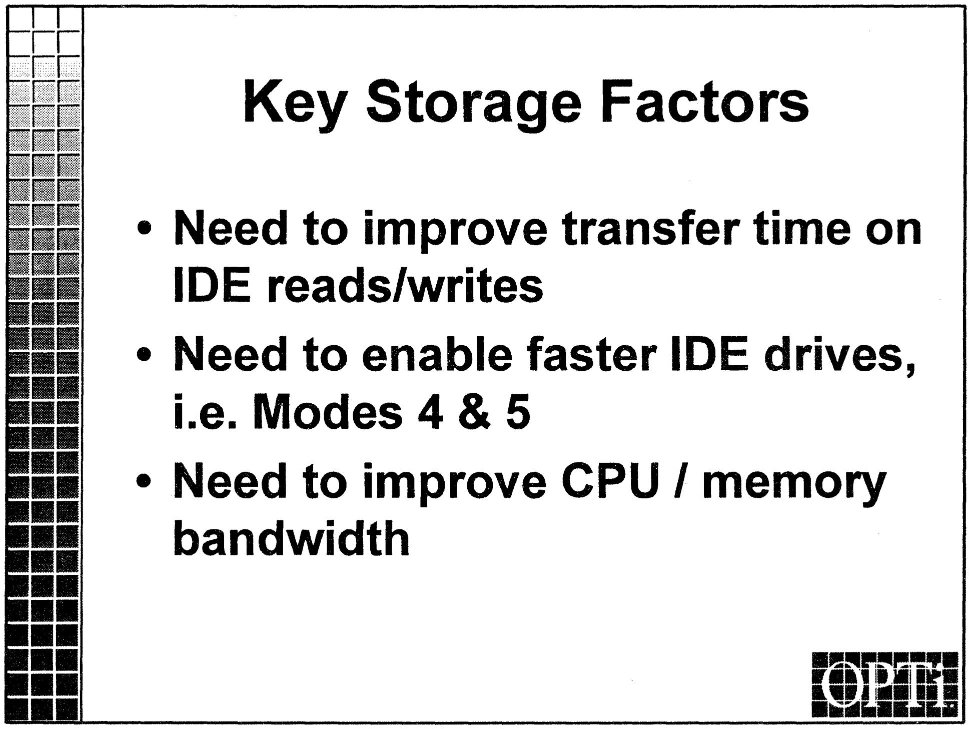
Sys Cntrlr
ISA Bridge

ISA bus



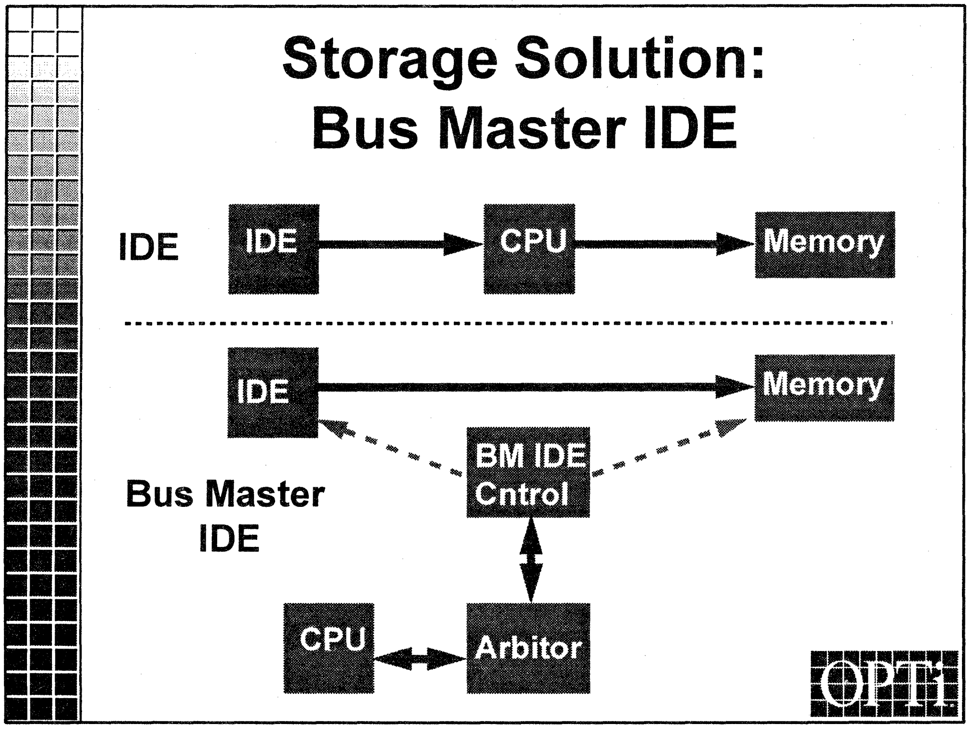
Audio Solution: Type F DMA

- Burst operation
- Cuts average DMA transfer from 1us to 250ns
- Decreases required CPU bus bandwidth from 20% to < 5%
- Requires FIFO in audio controller



Key Storage Factors

- Need to improve transfer time on IDE reads/writes
- Need to enable faster IDE drives, i.e. Modes 4 & 5
- Need to improve CPU / memory bandwidth



Storage Solution: Bus Master IDE

IDE

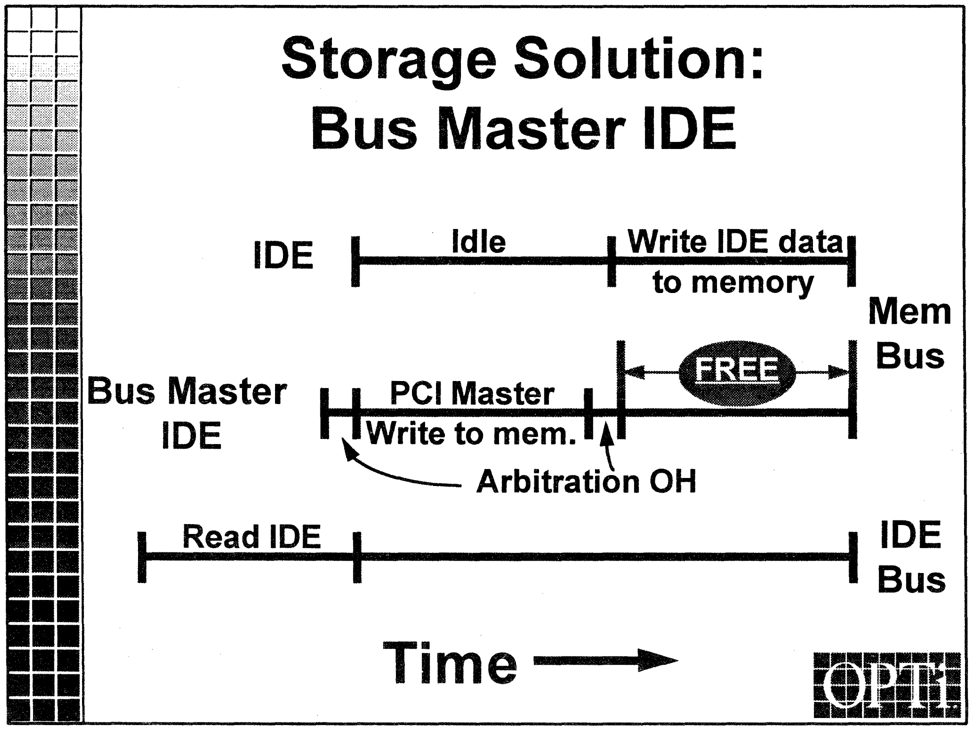
IDE → CPU → Memory

Bus Master IDE

IDE → Memory

BM IDE Cntrol

CPU ↔ Arbitor



Storage Solution: Bus Master IDE

IDE

Idle

Write IDE data to memory

Mem Bus

FREE

Bus Master IDE

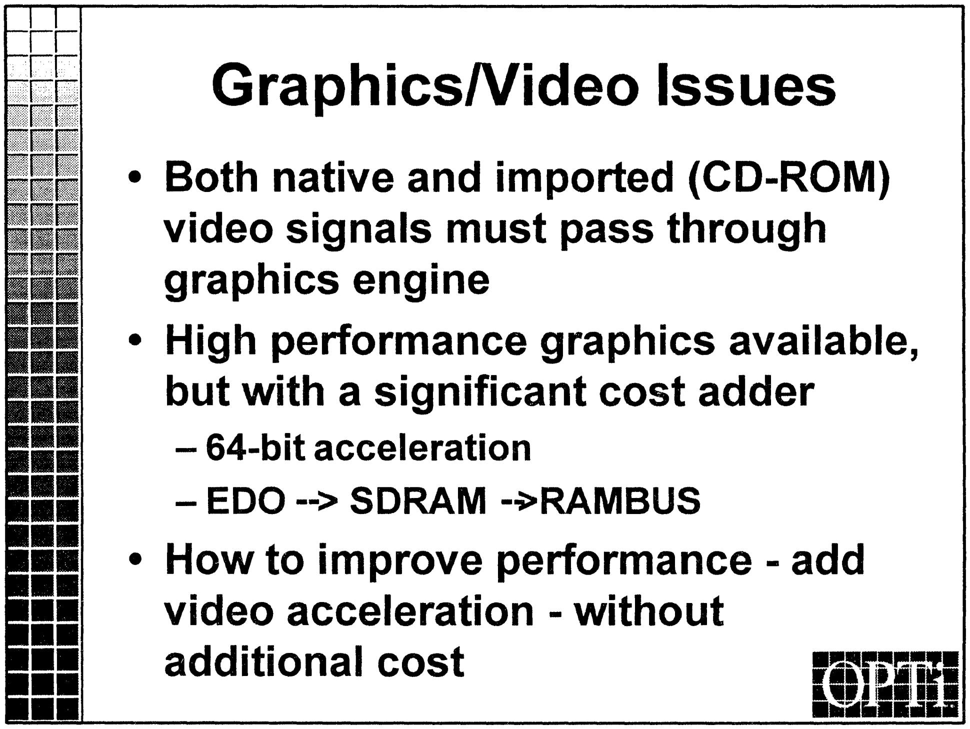
PCI Master Write to mem.

Arbitration OH

Read IDE

IDE Bus

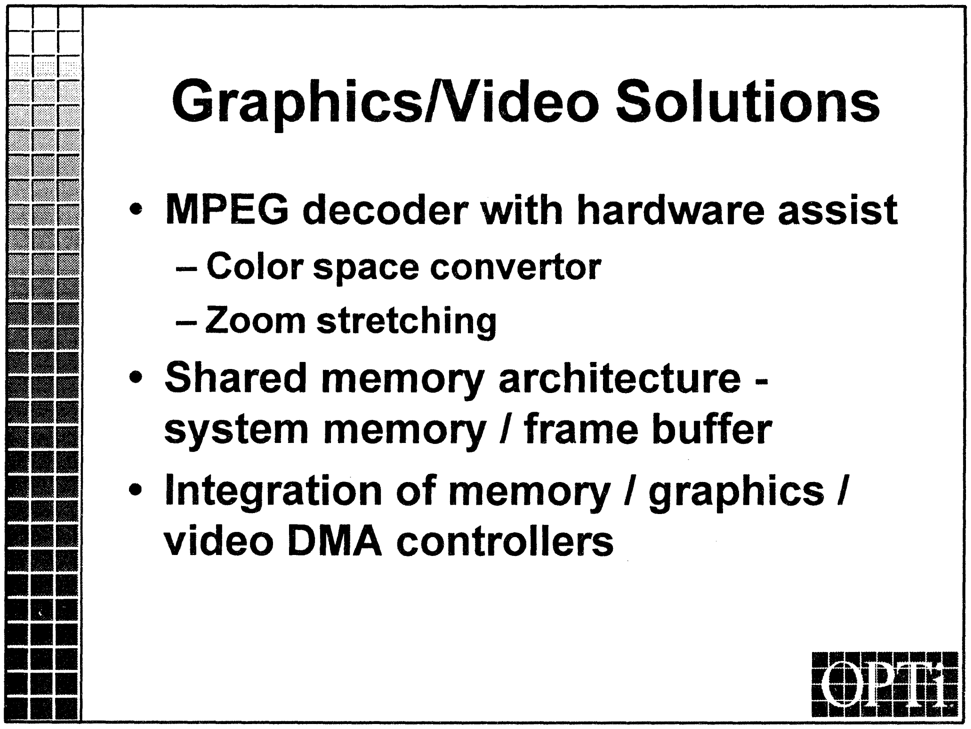
Time →



Graphics/Video Issues

- Both native and imported (CD-ROM) video signals must pass through graphics engine
- High performance graphics available, but with a significant cost adder
  - 64-bit acceleration
  - EDO --> SDRAM ->RAMBUS
- How to improve performance - add video acceleration - without additional cost

OPTi

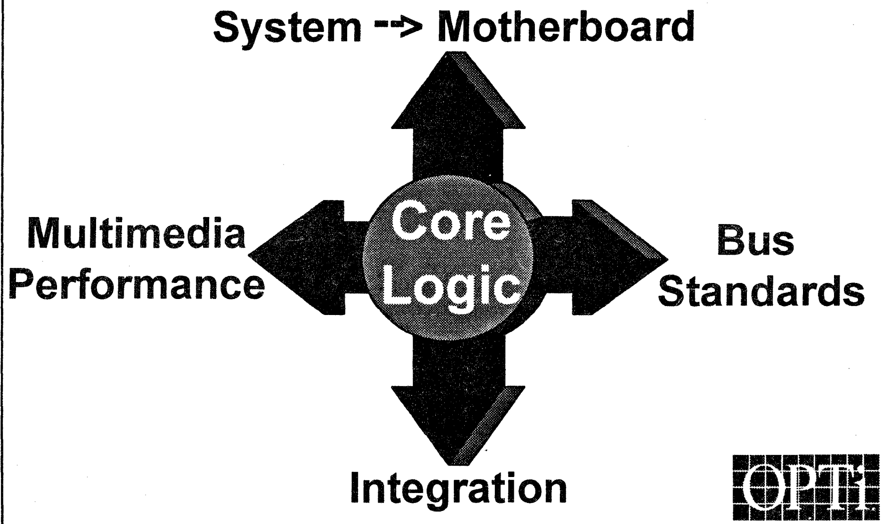


Graphics/Video Solutions

- MPEG decoder with hardware assist
  - Color space convertor
  - Zoom stretching
- Shared memory architecture - system memory / frame buffer
- Integration of memory / graphics / video DMA controllers

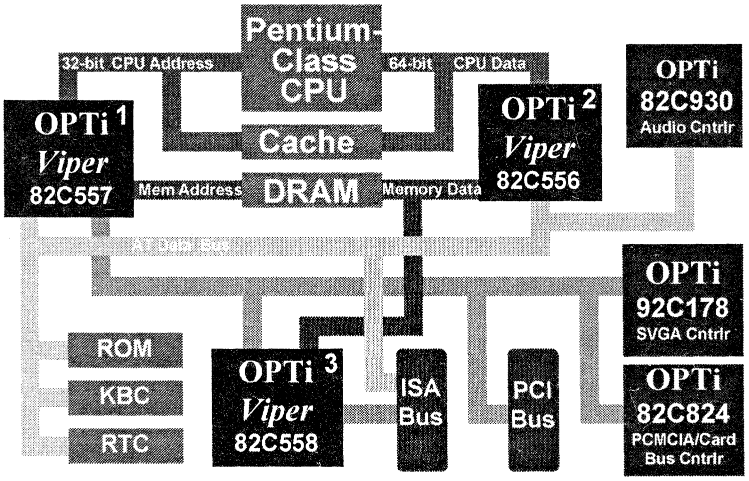
OPTi

The Role of Core Logic in Multimedia PCs



OPTi

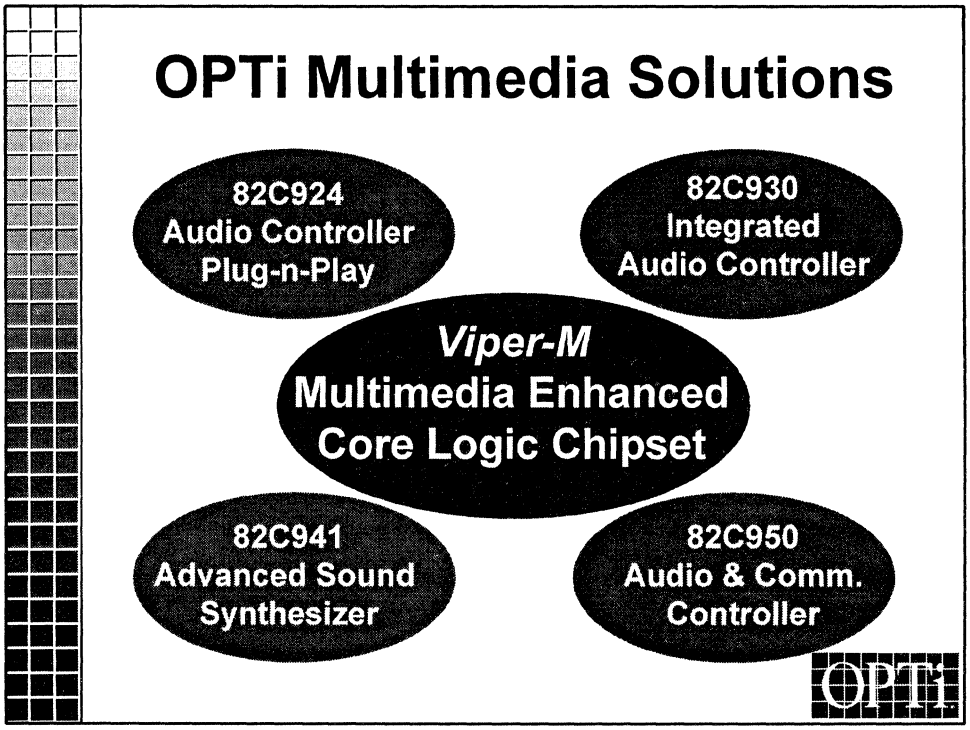
Pentium-Class System



1-System Controller
2-Data Buffer

3-Bus Controller/
Power Mgmt

OPTi



OPTi Multimedia Solutions

82C924
Audio Controller
Plug-n-Play

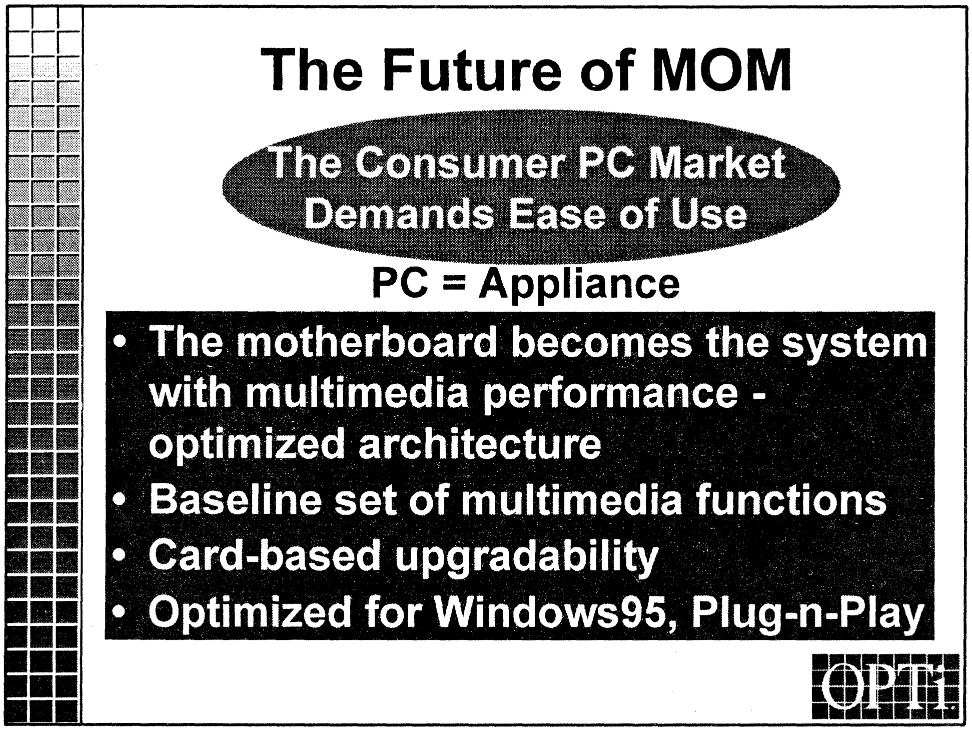
82C930
Integrated
Audio Controller

Viper-M
Multimedia Enhanced
Core Logic Chipset

82C941
Advanced Sound
Synthesizer

82C950
Audio & Comm.
Controller

OPTi



The Future of MOM

The Consumer PC Market
Demands Ease of Use

PC = Appliance

- The motherboard becomes the system with multimedia performance - optimized architecture
- Baseline set of multimedia functions
- Card-based upgradability
- Optimized for Windows95, Plug-n-Play

OPTi

PNP DOS*/Windows* Overview/Update

Scott Hay
Technical Marketing Engineer
Intel Corporation
5200 NE Elam Young Pkwy., HF3-64
Hillsboro, OR 97124
Ph. (503) 264-2217 Fax (503) 264-7902
March 29, 1995

-Plug and Play Association-

PNP DOS*/Windows* Overview/Update

Scott Hay
Technical Marketing Engineer
Intel Corporation
5200 NE Elam Young Pkwy., HF3-64
Hillsboro, OR 97124
Ph. (503) 264-2217 Fax (503) 264-7902
March 29, 1995

*Other trademarks are the property of their respective owners.

2/08/95

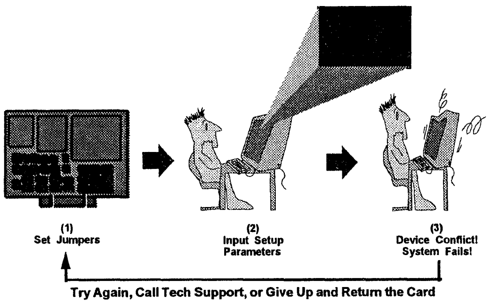
Agenda

- PnP Overview
- DOS*/Windows*/BIOS PnP Architecture
- PCMCIA on the desktop
- PnP DOS*/Windows* migration to Windows*95
- PnP Debug/Support Issues
- DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

2/08/95

The Desktop Add-in Problem


(1)
Set Jumpers
(2)
Input Setup Parameters
(3)
Device Conflict! System Fails!
Try Again, Call Tech Support, or Give Up and Return the Card

2/08/95

Solution

- Plug and Play enabled platforms
  - PnP BIOS
- Plug and Play add-in cards
  - PCI
  - Plug and Play ISA
  - PCMCIA
- Plug and Play enabled operating systems
  - DOS*/Windows*
  - Windows*95

Making the PC an Information Appliance

*Other trademarks are the property of their respective owners.

2/08/95

What is Plug and Play?

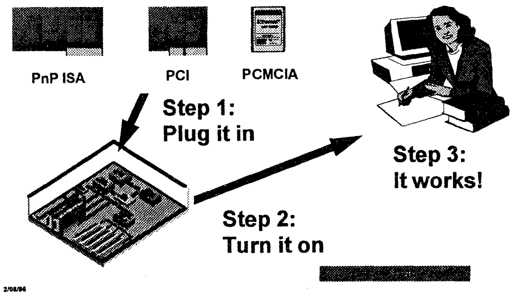
- Automatic configuration of add-in cards
- Major element of ease of use
- Industry backed standard
- Plug and Play is ready today on MS-DOS* and Windows* 3.1
  - Migratable to Windows*95
- Windows*95 product offering full PnP
- Industry well on its way to full implementation
  - 30+ OEMs & 70+ IHVs in development
  - 15+ OEMs and 20+ IHVs announced and shipping

Other brands and names are the property of their respective owners.

2/08/95

Plug and Play

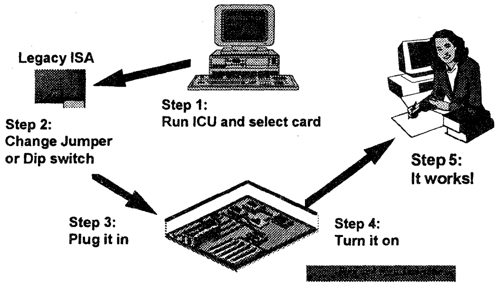
Plug and Play System with Plug and Play Cards



2/08/95

Plug and Tell

Plug and Play system and Legacy cards

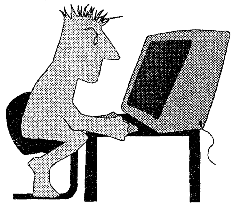


2/08/96

Plug and Hope

Legacy system and Legacy cards

- Today's situation
- No guarantee you will get the right configuration
- Trial and error



2/08/96

Plug and Play Association

- Administers Specifications
  - Plug and Play ISA specification 1.0A
    - ↗ Clarification Documents
  - Plug and Play BIOS specification 1.0A
    - ↗ Clarification Documents
  - Hosts Interoperability Workshops
    - ↗ 3 PlugFests to date
    - ↗ Upcoming event in April
  - 300+ members of the association
    - ↗ Open to all industry participants
    - ↗ $500/company annual fee

2/08/96

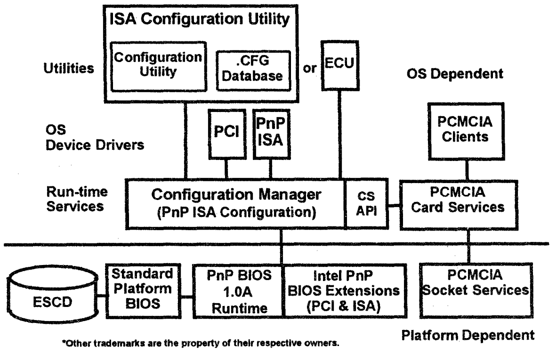
Agenda

- PnP Overview
- DOS*/Windows*/BIOS PnP Architecture
- PCMCIA on the desktop
- PnP DOS*/Windows* migration to Windows*95
- PnP Debug/Support Issues
- DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

2/08/96

PnP DOS/Windows* Architecture



*Other trademarks are the property of their respective owners.

Agenda

- PnP Overview
- DOS*/Windows*/BIOS PnP Architecture
- PCMCIA on the desktop
- PnP DOS*/Windows* migration to Windows*95
- PnP Debug/Support Issues
- DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

2/08/96

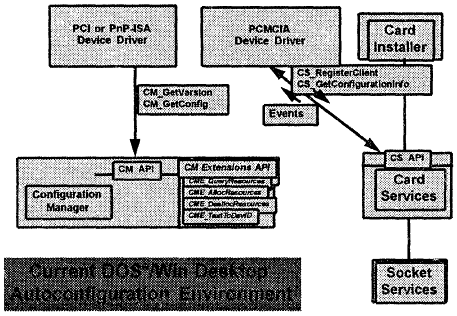
PCMCIA Autoconfiguration Today

- Card Services or Card Installer Device Drivers performs the autoconfiguration
  - ◆ Determines available system resources and assigns available requested resources to PCMCIA cards
  - ◆ Works ok in a mobile environment where limited peripheral/add-in card expansion exists
- Model breaks down in a desktop environment
  - ◆ CS/CI doesn't know how ALL the desktop resources have been assigned
  - ◆ CS/CI can't assign resources in a non-conflicting way

Desktop PCI & PnP ISA Autoconfiguration Today

- PnP BIOS
  - ◆ Autoconfigures PCI and PnP ISA add-in devices
  - ◆ Provides PnP runtime services
- PnP Software
  - ◆ DOS/Windows Kit available NOW from Intel
    - ↗ Configuration Manager
      - ↓Autoconfiguration of PnP ISA devices
      - ↓Provides API for PnP device drivers and software utilities
    - ↗ ISA Configuration Utility
      - ↓Integration guidance for legacy add-in cards
      - ↓Essential for accurate map of static resource usage
  - ◆ Microsoft Windows*95
    - ↗ Autoconfiguration of PnP devices

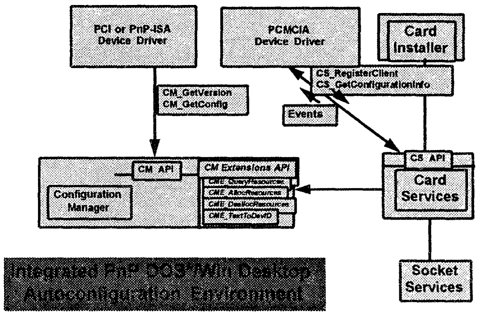
*Other trademarks are the property of their respective owners.



Current DOS*/Win Desktop Autoconfiguration Environment

PnP DOS*/Windows* and PCMCIA CS assign resources independently.

*Other trademarks are the property of their respective owners.



Integrated PnP DOS*/Win Desktop Autoconfiguration Environment

Config Manager Extensions (CME) for PCMCIA Integration

Integrated PnP Solution Available Today

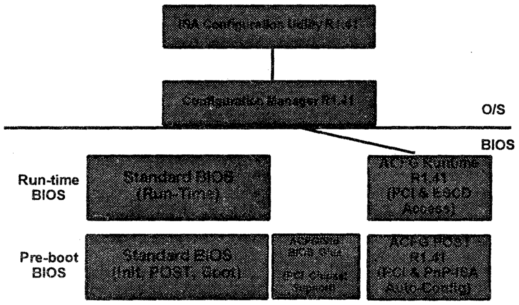
- Plug and Play DOS/Windows Kit Release 1.41
  - ◆ Autoconfigures PCI & PnP ISA cards
  - ◆ Provides a Configuration Manager API for Card Services
    - ↗ CS uses CM as a resource manager
    - ↗ CM provides resource map of all legacy, PCI and PnP ISA devices installed in the system
    - ↗ CS can now accurately and reliably assign resources to PCMCIA clients by calling CM
- Card Services Providers
  - ◆ Intel has informed all CS providers regarding the PCMCIA functionality supported by the PnP software kits
  - ◆ Card Services Providers
    - ↗ Developing CS capable of making the CM API calls
    - ↗ Card Installers can call CM or continue to call CS

Agenda

- PnP Overview
- DOS*/Windows*/BIOS PnP Architecture
- PCMCIA on the desktop
- PnP DOS*/Windows* migration to Windows*95
- PnP Debug/Support Issues
- DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

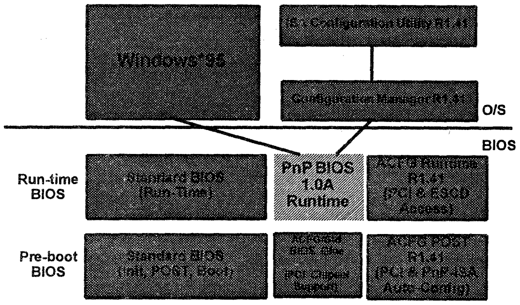
PnP BIOS for MS-DOS* and Windows* 3.1


ISA Configuration Utility R1.41
Configuration Manager R1.41
O/S
BIOS
Run-time BIOS
Standard BIOS (Run-Time)
ACFG Runtime R1.41 (PCI & ESCD Access)
Pre-boot BIOS
Standard BIOS (Init, POST, Boot)
Advanced BIOS Glue (PCI Chipset Support)
ACFG POST R1.41 (PCI & PnP-ISA Auto-Config)

*Other trademarks are the property of their respective owners.

2/08/96

PnP BIOS for Windows*95 or MS-DOS* and Windows*3.1


Windows*95
ISA Configuration Utility R1.41
Configuration Manager R1.41
O/S
BIOS
Run-time BIOS
Standard BIOS (Run-Time)
PnP BIOS 1.0A Runtime
ACFG Runtime R1.41 (PCI & ESCD Access)
Pre-boot BIOS
Standard BIOS (Init, POST, Boot)
Advanced BIOS Glue (PCI Chipset Support)
ACFG POST R1.41 (PCI & PnP-ISA Auto-Config)

*Other trademarks are the property of their respective owners.

2/08/96

Windows*95 PnP Architecture

- ◆ Bus enumerators present bus resource requests to CM
  - ↗ ISA, PCMCIA, PCI, BIOS, COM, Video, SCSI, EISA ...
- ◆ Arbitrators allocate/release resources
  - ↗ IRQ. DMA. Memory and I/O
- ◆ Device drivers support dynamic events
  - ↗ Insertion/removal/docking
  - ↗ Suspend/resume
- ◆ CM performs resource balancing
- ◆ Device manager reflects system resources present
  - ↗ Power users, shielded from most users
- ◆ Need to develop specific Windows 95 device driver

*Other trademarks are the property of their respective owners.

2/08/96

Agenda

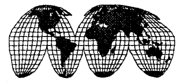
- ▪ PnP Overview
- ▪ DOS*/Windows*/BIOS PnP Architecture
- ▪ PCMCIA on the desktop
- ▪ PnP DOS*/Windows* migration to Windows*95
- ▪ PnP Debug/Support Issues
- ▪ DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

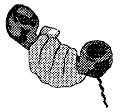
2/08/95

PnP as a Support Tool

- ▪ Use the ICU to install legacy cards!
  - ◆ ESCD must describe all static resources
  - ◆ ESCD is a resource map



- ▪ Use the ICU as a tool for on-line debugging



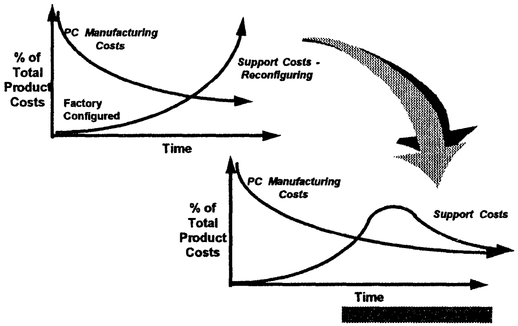
2/08/96

PnP DOS/Windows* Support Issues

- ▪ Customer doesn't use ICU
- ▪ PnP Learning curve
  - ◆ Calls may go up initially
  - ◆ Customer may be inclined to install legacy card before running ICU (DON'T LET THIS HAPPEN!)
- ▪ Customer still needs to load device drivers
  - ◆ Windows*95 will dynamically load DD
- ▪ Multiple function card not completely configured
  - ◆ ISA limitation: May still run out of available resources
    - ↗ IRQ, DMA, I/O, Memory

2/08/96

Plug and Play Saves Money



PnP Issues Continued

- Non-PnP card claiming to be "Plug and Play"
- Legacy card with no CFG file for ICU
  - ◆ Add CFG file
  - ◆ Use "unlisted" option in ICU
  - ◆ Contact Intel to develop or integrate CFG
- Other issues?

Agenda

- PnP Overview
- DOS*/Windows*/BIOS PnP Architecture
- PCMCIA on the desktop
- PnP DOS*/Windows* migration to Windows*95
- PnP Debug/Support Issues
- DOS*/Windows* Product Plans/Support

*Other trademarks are the property of their respective owners.

Plug and Play Kit Plans

- Release 1.41JP Japanese (Kanji) Win 3.1 (no DOS)
- Release 1.41CN Chinese Win 3.1 (no DOS)
  - ◆ Goal: March/April
- Release 1.41 BIOS Enhancement and DOS/Win 3.1
  - ◆ Release: February
  - ◆ Feature enhancement: PCI-PCI bridge support
  - ◆ 10 European Languages Support
    - ↗ Goal: March

Future PnP Software Products

- Intel will continue to update software kit as needed
- Symantec incorporated ICU/CM in "More PC-Tools" (Q4/95)
  - ◆ Available in retail stores
- Microsoft Windows*95
  - ◆ Rich Ease of Use environment
- IBM OS/2
  - ◆ Incorporating Ease of Use features

PnP Support Contacts

- Customer Support: 1-800-628-8686 or 1-916-356-3551
- Customer FaxBack: 1-800-628-2283
- BBS: 1-916-356-3600 North America
  44-793-496340 Europe
- Developers Kit: 1-800-253-3696
- Plug and Play Association Information
  P.O. Box 14070
  Portland, OR 97214-9499
  (800) 433-3695, (503) 797-4244
  (503) 234-6762 Fax

PnP Support Contacts cont.

- **Compuserve Forum: "go plugplay"**
  - ◆ PnP ISA, BIOS, SCSI, Mobile, PCMCIA etc.
- **Internet**
  - ◆ ftp.intel.com or www.intel.com
  - ◆ ftp.microsoft.com or www.microsoft.com

2/08/96

Summary

- Plug and Play is:
  - ◆ Developed for MS-DOS* and Windows* 3.1 today
  - ◆ Fully compatible with Windows*95
- Plug and Play saves money, reduces end-user frustration
- Ship systems with ICU/CM pre-installed
  - ◆ ESCD reflects all cards including legacy
- Plug and Play is an industry backed standard
- OEM's, BIOS and IHV's have begun rolling out Plug and Play Products now

**Other brands and names are the property of their respective owners.*

2/08/96

Call To Action

- Focus on making the PC as easy to use as possible
- Develop PCI cards fully compliant to the PCI spec.
  - ◆ Assures full PnP functionality
- Develop fully integrated PnP PCs
  - ◆ PCI
  - ◆ PnP ISA
  - ◆ PCMCIA

Make the PC an Information Appliance

2/08/96

Key Message

- **To get Plug and Play today you must have:**
  - ◆**Plug and Play BIOS**
    - ↗Compliant to 1.0A specification
  - ◆**CM and ICU installed on hard disk for MS-DOS* and Windows* 3.1 systems**
  - ◆**Plug and Play cards**
    - ↗PCI, PnP ISA or PCMCIA

* Other trademarks are the property of their respective owners.

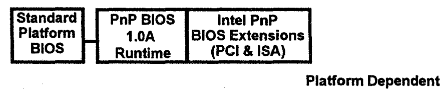
2/08/96

Appendix

2/08/96

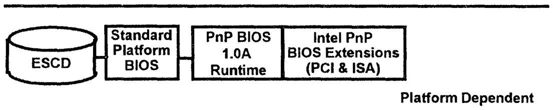
PnP BIOS

- Configure Plug and Play devices
  - ◆ Detect
  - ◆ Allocate system resources
  - ◆ Configure
- Run-time Access to Configuration Information
  - ◆ Motherboard
  - ◆ ISA, EISA, MCA
  - ◆ Plug and Play ISA and PCI



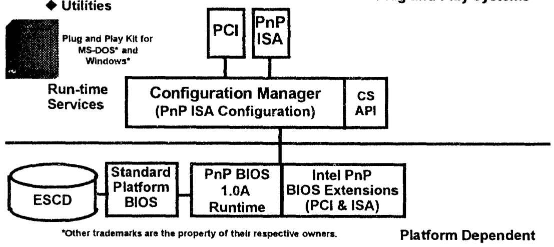
Extended System Configuration Data

- Stores All Configuration Information
  - Motherboard devices
  - ISA, EISA, MCA add-ins
  - Plug and Play add-ins
- Extends EISA Configuration Structure
- Plug and Play Configuration Information
  - Last working configuration
    - Previous boot assigned resources
    - ICU and ECU assigned resources
  - Locked Resources
    - Non P&P environments (i.e. Unix)



Configuration Manager

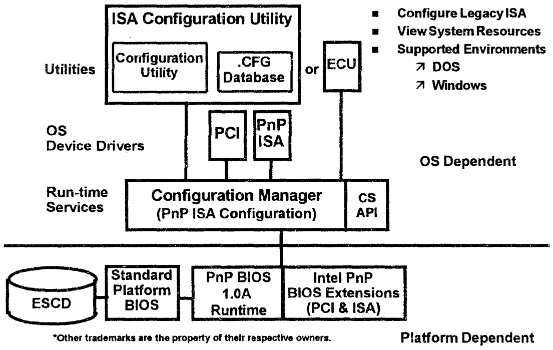
- Configure Plug and Play ISA
  - ◆ Legacy systems
- Access to configuration data
  - ◆ Device drivers
  - ◆ Utilities
- Assist PCMCIA Card Services
- Supported Environments
  - Legacy systems
  - Plug and Play systems


Plug and Play Kit for MS-DOS* and Windows*
PCI
PnP ISA
Run-time Services
Configuration Manager (PnP ISA Configuration)
CS API
ESCD
Standard Platform BIOS
PnP BIOS 1.0A Runtime
Intel PnP BIOS Extensions (PCI & ISA)
Platform Dependent

*Other trademarks are the property of their respective owners.

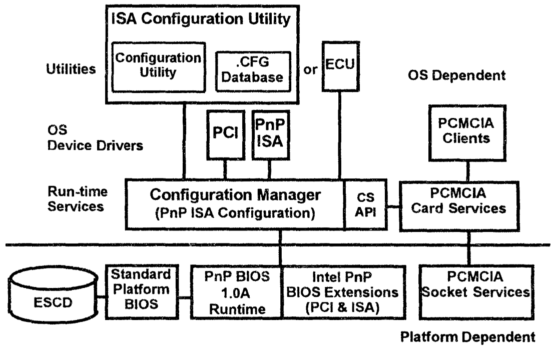
ISA Configuration Utility

- Configure Legacy ISA
- View System Resources
- Supported Environments
  - ↗ DOS
  - ↗ Windows

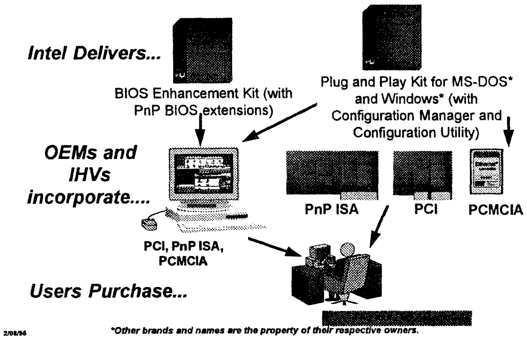


*Other trademarks are the property of their respective owners.

PnP Architecture



Plug and Play Implementation Today


Intel Delivers...
BIOS Enhancement Kit (with PnP BIOS extensions)
Plug and Play Kit for MS-DOS* and Windows* (with Configuration Manager and Configuration Utility)
OEMs and IHVs incorporate....
PnP ISA
PCI
PCMCIA
PCI, PnP ISA, PCMCIA
Users Purchase...

2/08/96

*Other brands and names are the property of their respective owners.

ABSTRACT

Obtaining Maximized Performing Cost-Efficient Design With Core Logic for Pentium-based PCI Systems

by Dr. S.J. Lee
Acer Laboratories Inc. (ALi)

Ali (Acer Laboratories, Inc.) will discuss the key issues in providing PCI Pentium PC designers with system core logic which is able to meet the system cost-pressures of Pentium-class CPUs becoming a main-stream technology, without sacrificing performance.

The product with which Ali has implemented its solution is the Aladdin M1511 Memory Buffer Controller, M1513 System I/O Controller and M1512 Data Path Buffers.

One of the key elements in the chip architecture which Ali has designed for its Pentium/M1/K5 system core logic is its buffering approach to maximize data transfer and maintain concurrent operations between CPU, memory, PCI bus and IDE. The Memory Buffer Controller integrates the cache controller, memory controller and the buffer controller between Host and PCI. The buffer controller is used to optimize the Host to PCI memory cycle to improve the graphic performance, by merging the Host byte/word/dword cycle to perform burst or back-to-back PCI cycles. The memory controller is architected to support memory sizes up to 768 MB in six banks. The secondary level cache can be 256K, 512K or 1MB in write-back mode, using asynchronous or Pipeline Burst SRAM performing an N-1-1-1 cycle are all supported in Aladdin. Fast page mode DRAMs and EDO DRAMs performing an N-2-2-2 cycle can be achieved for maximum system design flexibility and cost optimized design.

The 8-level qword write buffers to 64-bit main memory in the Data Path buffer, have been designed to quickly respond to memory requests of both Host and I/O Masters. Posted write buffer and prefetching read buffer between Host and PCI bus have been developed to not only sustain the Host to PCI slave throughput, but also to maximize the utilization of PCI bandwidth requested by PCI masters. The APIC multiprocessor protocol and all PC-AT macro logic are supported in the System I/O controller. Also, the IDE interface controller and AT Keyboard controller are integrated, providing greater cost efficiency and design options.

ALTA-S/MP MEMORY CONTROLLER AND PCI BRIDGE

John Derrick
IBM Microelectronics
1000 River Street, Essex Jct., VT 05452

ABSTRACT

ALTA-S/MP is a family of memory controller and PCI Bridges that are optimized for peak PCI performance. They have been architected to maximize the bandwidth of the PCI Bus in real system design.

Extensive prefetching and buffering of reads and writes to main memory, PCI memory address space, and other innovative features that optimize bandwidth provide leading edge performance while providing full processor and PCI bus parity and memory EDAC protection.

These leading edge performance enhancements and programmable features also enable the ALTA-S/MP family to support processors from Intel, Cyrix, IBM, and other similar processors in an optimal fashion.

IBM's leading edge device, packaging, and tools technology enabled the ALTA-S/MP design team to implement 133 MHz I/O resolution for memory control signals and 133 MHz internal logic to maximize system performance.

KEY FEATURES

Processors Supported:

Intel Pentium$_{(tm)}$ Class Processors
Cyrix 586-Class Processors
IBM 586-Class Processors
AMD 586-Class Processors

DRAM Interface

*Up to 1 GB of ECC-Protected DRAM.
*ECC Codes are generated/checked with Parity Performance.
*Single Bit and Double Bit Errors are recorded, Double Bit errors are tagged in memory so no error information is lost.
*16 Non-Interleaved 64-bit or 8 Interleaved 128-bit Banks.
*1/2 Clock resolution RAS and CAS control pulses for optimal memory timings. (7.5 ns at 66 MHz)
*Partial Write Optimization minimizes reads for optimal ECC read-modify-write performance.
*Page_Miss_Mode to eliminate Page Miss Penalty in Dual and Multi-processor Systems.
*Address and Data-Flow paths are latched at chip edge thus, timings are predictable allowing for optimal system designs.
*Memory Map is highly programmable with 4 MB resolution of the memory map in non-interleave mode and 8 MB resolution in interleave mode.

CPU Interface

*Supports Serial and Look-Aside L2 Caches with CPU Address Pipeling and Programmable Cache Timings.
*Full CPU Address and Data Parity Generation and Checking.
*Extensive Buffering and Prefetching

PCI Master Interface

*Programmable ROM Decode
*Write Buffering with Programmable Compression
*PCI 2.0 Compliant, will be PCI 2.1 Compliant

PCI Slave Interface

*Buffering and Prefetching
*PCI 2.0 Compliant, will be PCI 2.1 Compliant

KEY PERFORMANCE ADVANTAGES

ALTA-S/MP offer significant performance advantages not found in other memory controllers and PCI bridges. They also support a wide range of processors and external cache controllers to match the price performance required for a wide range of system needs. These features are outlined below.

Level-2 Cache Controllers

The ALTA-S/MP family of memory controller and PCI Bridges support both a Look-Aside Cache and various Serial Cache Controllers.

Look-Aside LYNX L2 Cache Controller (Write-Back or Write-Thru)

LYNX using Async SRAM can support 256K, 512K, 1M & 2MB L2 Caches.
3-2-2-2 bursts when running without CPU Address Pipeling.
(about 213 MB/sec local bus bandwidth @ 66 MHz)
3-2-2-2-2-2-2-2 bursts when CPU Address Pipeling is enabled.
(approaches 267 MB/sec local bus bandwidth @ 66 MHz)

LYNX using Sync SRAM can support 256K, 512K & 1MB L2 Caches.
3-1-1-1 bursts when running without CPU Address Pipeling.
(about 305 MB/sec local bus bandwidth @ 66 MHz)
3-1-1-1-2-1-1-1 bursts when CPU Address Pipeling is enabled.
(about 427 MB/sec local bus bandwidth @ 66 MHz)

Serial L2 Cache Controllers (Write-Back or Write-Thru)

ALTA-S/MP supports both a local bus frequency matching the CPU local bus frequency and a local bus frequency matching the PCI bus frequency. This allows the most flexibility in choosing Serial L2 controllers. CPU Address Pipeling is fully supported for serial caches also.

Full Concurrency:

The following groups of transactions may occur simultaneously. Interrupt Acknowledge cycles and I/O cycles cause the write and pre-fetch buffering to be cleared.

CPU-to-L2 Accesses
CPU-to-PCI Accesses
CPU-to-Memory Accesses
PCI-to-Memory (non-Cacheable) Accesses

CPU-to-L2 Accesses
CPU-to-PCI Accesses
PCI-to-Memory (Cacheable) Accesses

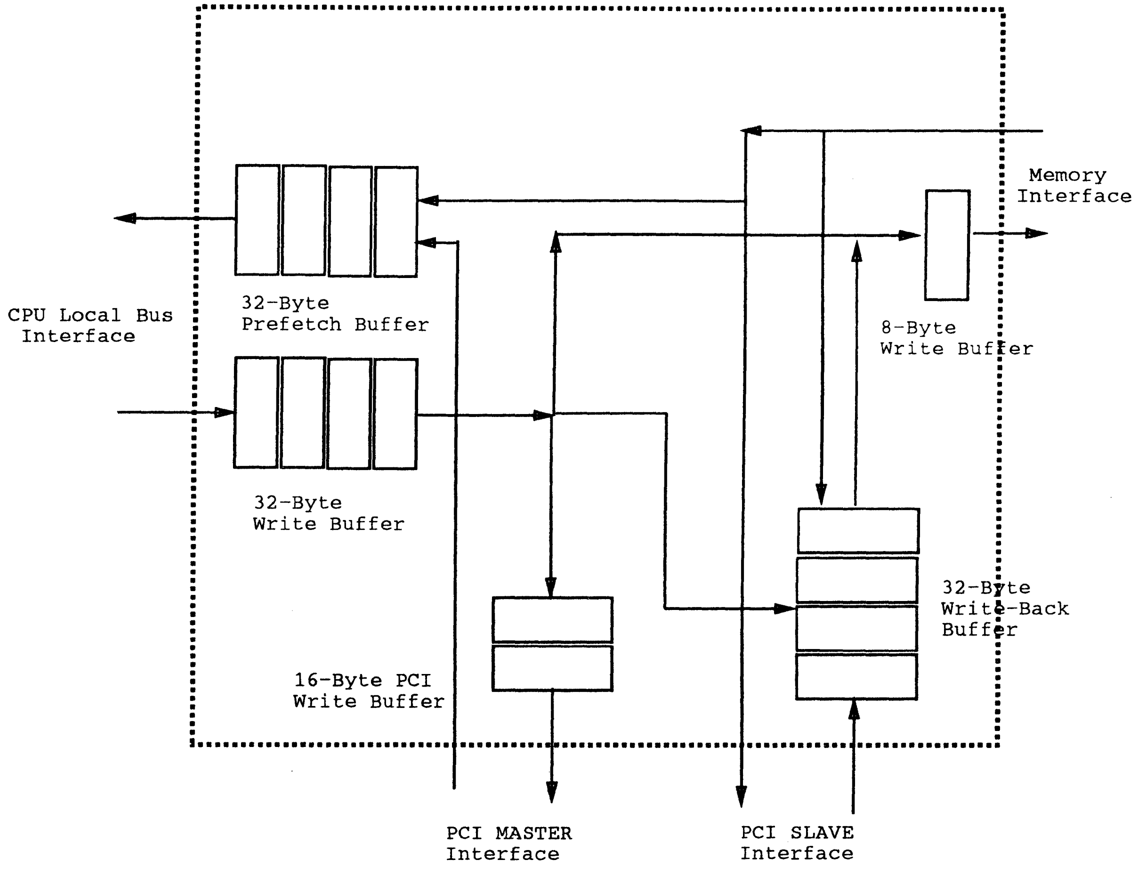
The CPU-to-Memory Data write path includes a 40-Byte write buffer for cast-outs and single memory writes.The write-back data path for PCI originated snoops have 32-Bytes of CPU frequency buffering and 32-Bytes of PCI frequency buffering.

The CPU-to-Memory Data read path includes a 32-Byte read buffer used for prefetched read allocation cycles during cast-out for allocation cycles. This allows data to be retrieved for a line-fill concurrent to the L2 casting out the dirty cache line about to be replaced.

The CPU-to-PCI Data write path includes a 24-Byte write buffer that offers programmable compression. This buffering and compression enables the processor to generate higher bandwidth (burst transfers) as the PCI bus becomes utilized. This offers some dynamic performance tuning without adding latency to CPU-to-PCI write cycles.

The PCI-to-Memory Data write path includes a 32-Byte Buffer, shared with the write-back data path, that allows PCI masters to write a full cache-line of data concurrent with the processor snoop inquire or snoop invalidate cycle.

ALTA-S/MP BUFFER SCHEME



Concurrent Operations:

The 16-Byte PCI Write Buffer may contain memory write cycles and data destined for PCI Address Space.

The 8-Byte Write Buffer to Memory may contain a memory write cycle destined for Planar Address Space.

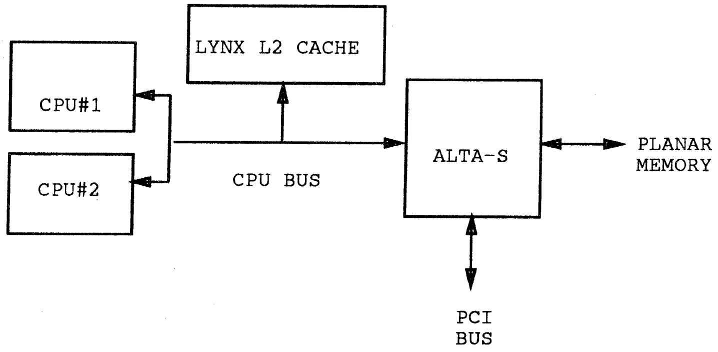
The 32-Byte Write-Back Buffer may be prefetching a PCI Master memory read or posting a PCI Master memory writes.

The 32-Byte Write Buffer may be posting the Cache Write-Back while the 32-Byte Write-Back Buffer is prepared for data merge.

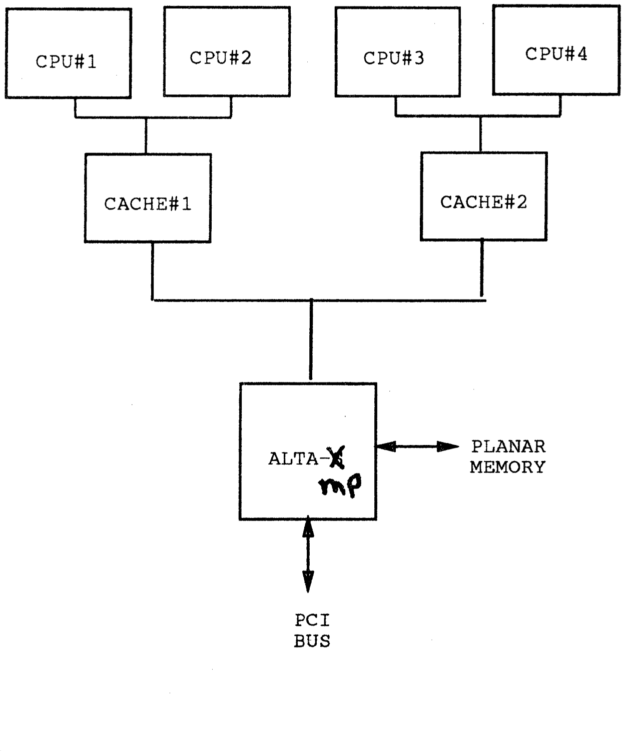
The 32-Byte Write Buffer may fill while the 32-Byte Prefetch Buffer fills during a Cast-Out for Allocation.

note: Locked Accesses and Interrupt Acknowledge Cycles cause buffers to flush.

ALTA-S



ALTA-MP



Overview of the use of the PCI bus in Present and Future High Energy Physics Data Acquisition Systems

A. van Praag, R.A. McLaren, J-P. Matheys, P. Vande Vyvre, CERN, Geneva, Switzerland

T. Anguelov, G. Georgiev, S. Piperov, I. Vankov, INRNE - BAS, Sofia, Bulgaria

D. Gillot, A. Guglielmi, Digital Equipment Corporation, Joint project office, CERN

O. Orel, A. Sytin, IHEP, Prodvino, Russia

ABSTRACT

Due to its very complex data acquisition systems High Energy Physics (HEP) experiments are always looking for cheap and fast computers and communication equipment. PCI as a mainstream product is one of the new technologies responding to these criteria. After a short introduction of CERN and its Particle Physics Facilities, the first part of this article describes, with a real development project as example, the specific problems of data acquisition HEP experiments with the future LHC accelerator. Solutions where PCI technology will play a role will be presented, showing as examples the use of a VMEbus module with dual port ram and PCI to SCI interfaces. The second part describes the NA48 experiment including a detailed description of the development of the PCI to HIPPI interface.

INTRODUCTION

In the war-ravaged Europe of the early 1950s, a far sighted group of scientists and politicians envisioned a new adventure in science, a European scientific laboratory. Even then, it was clear that state-of-the art science needed research facilities larger and more complex than individual nations could afford. In this way Europe's role in fundamental science would be restored, at the same time bringing together people from countries which had been at war only few years before. In 1954 twelve countries started to work on a 600 MeV Synchro-Cyclotron on the Meyrin Site in Switzerland. In Parallel CERN began to build the Proton Synchrotron (PS). This came into operation in 1959 and for a time was the most powerful particle accelerator in the world, supplying experiments with 28 GeV beams of protons. This accelerator is still the kingpin, being the first of the actual system of interconnected accelerators.

In 1976 the PS was followed by a new more powerful Super Proton Synchrotron (SPS) machine of 450 GeV. This accelerator has a circumference of 7 Km and passes under the Swiss-French border.

In 1983 work started on the Large Electron Positron Accelerator (LEP). Constructed in a 27 Km tunnel, the first particles were accelerated during 1989, with an energy of up to 45 GeV. LEP will by 1995 reach an energy of 90 GeV.

On 16 December 1994 the CERN member states decided to continue the extension of the laboratory with the construction of the Large Hadron Collider (LHC) in the same 27 Km tunnel, having two intersecting accelerator tubes with an energy of over 7 TeV each. During the short history of CERN the number of member-states has grown to nineteen and four more countries have an observer status.

PART 1: DATA ACQUISITION FOR THE LHC ATLAS DETECTOR

Several physics experiments will use the LHC accelerator. Three of them have already been approved: Atlas and CMS will study proton-proton collisions and ALICE will observe heavy ion interactions. An experiment consists of different specialized detectors each of them containing tens of thousands of channels. There are interactions in the detectors each 25 ns (40 MHz), but only a fraction of these is of interest. Filters are foreseen with several levels of triggers to select and store only selected data. In Atlas the architecture uses three levels (LVL1, LVL2, LVL3) as shown in Fig. 1. At LVL1, special-purpose processors act on reduced granularity data from a subset of the detectors. The LVL2 detector trigger uses full granularity, full precision data from most of the detectors, but examines only regions of the detector identified by LVL1 as containing interesting information, the so-called Regions Of Interest (ROI). At LVL3, the full event data is used to make the final selection of events to be recorded for off-line analysis. The LVL1 trigger accepts data at the full LHC bunch crossing frequency of 40 MHz (every 25 ns). The latency time necessary to form and distribute the LVL1 decision is ~2 μs, and the number of positive decisions is expected to be 10^5/s. Hence the LVL1 trigger must select no more than one interaction in ~10^4. During the LVL1 trigger processing the data from all parts of the detector are held in pipeline memories. Requirements of the LVL1 trigger are that it must identify unambiguously the bunch crossing that contains the region of interest with a negligible deadtime.

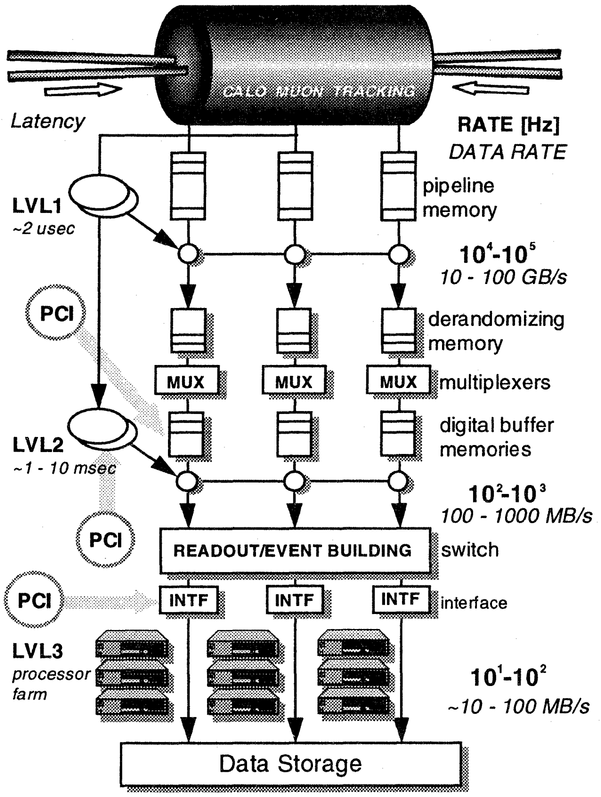


Fig 1: The Atlas Architecture

The LVL2 trigger must reduce the data rate from up to 100 KHz after LVL1 to ~1 KHz. Its architecture is based on ROIs. The LVL2 trigger has therefore to access and process only small fractions of the data which is an advantage for the required processing and data movement capacity. The processing is divided in two phases, extraction of the ROI and summarize it in a few data words, and combine it with information from other ROIs to make the LVL2 decision. The LVL2 latency is variable from ~1 to ~10 ms.

After an event is accepted by the LVL2 trigger, the full data is sent, via the event builder, to the LVL3 processor farm where reconstruction of the physics phenomena is possible. Decision times are up to ~1 s. After the final selection made by LVL3, data will be stored at a rate of 10-100 MB/s.

CAN PCI COMPONENTS BE OF USE?

The planning for LHC is that it will be operational in 2004, and that the technology will be frozen in 1997. Standards will be used wherever possible.

For the LVL1 part, the time available to do some very specific operations on the unfiltered data makes it necessary to construct the largest part of this front end with dedicated electronics.

The Digital Buffer Memories

Data accepted by the LVL1 trigger is transmitted to the digital buffer memories (Fig 1). Data input rates of 100 MB/s per memory are expected. Simultaneously, data from accepted LVL2 events must be output to the event builder. A dual port memory architecture with sophisticated memory management is therefore required. Currently

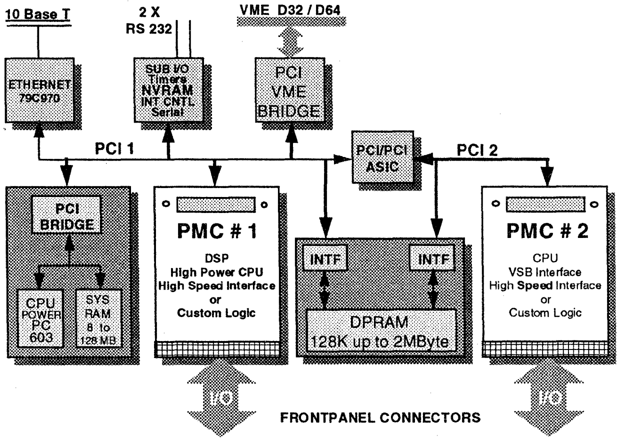


Fig 2: The RIO2 Block Diagram

neither the input nor the output links have been chosen. Therefore it could be judicious to specify the PCI bus as input and output to the digital memories, allowing easier implementation of any link type. An example of a suitable module is currently being designed by Creative Electronic Systems (CES). The RIO2 VMEbus module is shown in Fig 2. It has a Power-PC 603 or 604 as processor, 8 - 128 MB memory and a PCI main bus. One PCI Mezzanine Card (PMC) slot is directly coupled to the main bus. The second PMC is coupled to this bus both via a PCI to PCI buffer, and via an up to 2 MB dual port memory. The throughput of each memory port is 132 MB/s.

The LVL2 and LVL3 Interfaces

Referring again to Fig 1, the output of the digital buffer memories are transmitted through high speed links to the input of the event builder. The outputs of the event builder are connected to the LVL3 processor farm interfaces built out of standard workstations. Most manufacturers have announced PCI on their new models. The link technology used can be (Serial) HIPPI or in the near future Fibre Channel Standard (FCS) or Scalable Coherent Interface (SCI). ATM is another possibility under evaluation. If we assume that the digital buffer memory and the LVL3 interface use PCI internally, then interfaces between the link technology and PCI are required. Possible solutions for the system are:

1: HIPPI

The maximum throughput for HIPPI is 100 - 200 MB/s as given in the HIPPI-PH specification. Several HIPPI to PCI and Serial-HIPPI to PCI interfaces are under development by Genroco, Essential Communications and at CERN. No developments for the PMC form factor are known.

2: FCS

The maximum throughput for FCS is 100 MB/s as given in the FC-PH specification. Several PCI to Fibre Channel and PMC to Fibre channel interfaces are under development in industry by among others Western Digital, Emulex and Interphase. One of the difficulties is to combine the standard optical FCS modules with limited dimensions, especially height, of the PMC.

3: ATM

ATM specifies speeds from 155 Mb/s up to 2.4 Gb/s, where 1.2 Gb/s corresponds with 100 MB/s. PCI to ATM interfaces are under development by several companies, for example Newbridge, Efficient Networks and Digiboard. The development of an ATM PMC inetrface is under evaluation as part of a collaboration between at CERN and Uppsala University. All these interfaces cover the 155 Mb/s speed only.

4: SCI

The maximum link speed in the SCI specification is 1 GB/s.

A development project for a PCI to SCI interface has been started at CERN. The interface to the PCI bus will use the AT&T Orca FPGA, for data and PCI bus control, as shown in Fig 3.

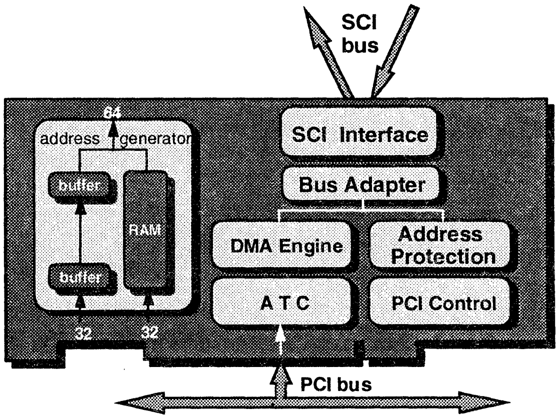


Fig 3: The CERN PCI to SCI Interface

The internal logic of the card is built with a DMA engine and address protection logic both implemented in PALs. They couple via a bus adapter to the Cbus entry of the SCI interface, a Dolphin Nodechip. The 64 bit SCI address is built by storing a number of the high words (A32-A63) in RAM and passing the low word (A00-A31), via the necessary buffers for bus adaptation, directly from the PCI bus to the SCI interface.

A very similar PCI-SCI interface but including a 256 word bi-directional FIFO in the data path is developed by Manchester University in collaboration with CERN [Hughes, 1994]. It is intended to be used in the Atlas LVL2 trigger.

PART 2: DATA ACQUISITION FOR NA 48

The NA 48 is a CP violation experiment. It is now in the construction phase. It should be fully operational in 1996 and finish before the installation of LHC.

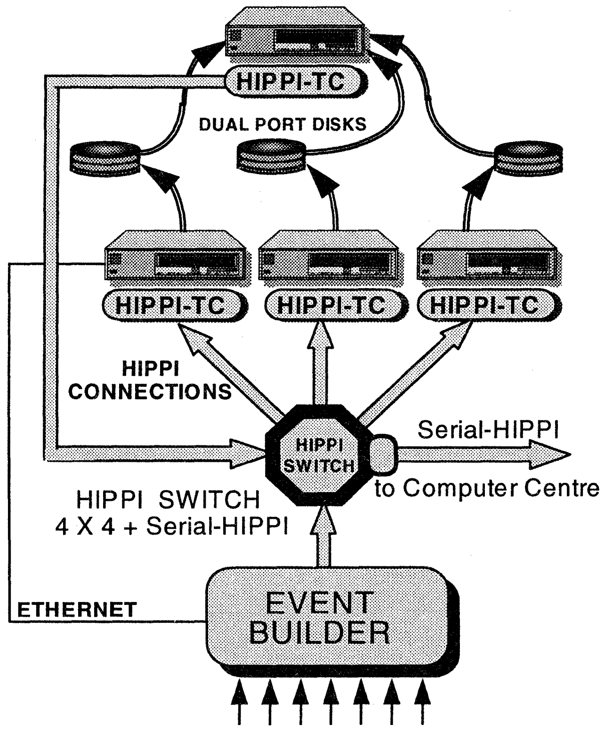


Fig 4: The NA 48 Data Flow

In the NA 48 experiment a few thousand events per second are expected after the second level trigger (LVL2) during every accelerator cycle (a spill of ~2.5 s every 15 s), resulting in a data block of up to 250 MB. These data are then to be processed by the third level trigger (LVL3) which requires the power and flexibility of powerful workstations. Three sequential interleaving workstations are used because processing of this quantity of data needs more time than is available between spills. The very high event rate does not allow any software intervention during the data transfer. Local disks are used for storage, complemented with a 10 Km high speed fiber optic link to the central computer center. Fig 4 shows an overview of the system.

Spill distribution is based on point-to-point HIPPI connections. The data block is distributed spill by spill using a HIPPI crossbar switch. The processors are DEC Alpha workstations with TURBOchannel interfaces. As part of a joint project, HIPPI to TURBOchannel interfaces were developed at CERN; the OSF/1 device drivers were developed by DEC. On top of the driver is a user level library. Tests with this architecture have shown that 64 MB of data can be transferred in ~770 ms from the event builder to a DECstation AXP 5000/200, including software overhead and Ethernet feedback to the data source. This corresponds to a transfer speed of 83.1 MB/s. Block sizes larger than 64 MB could not be tested because of the limits in the OSF/1 version 2 operating system. The very recent OSF/1 version 3 does not have this limit and tests with block sizes of up to 250 MB have been performed. To connect to the central computer center, one port of the switch is equipped with a Serial HIPPI module. The connection is made with single mode fiber optic cable.

Moving From TURBOchannel to PCI.

In the future more powerful workstations are needed. DEC has announced that new workstations will no longer support TURBO channel, instead they will be equipped with PCI interfaces.

In a new joint project the former partners have therefore agreed to develop a PCI to HIPPI interface that meets the NA48 specifications, adapting at the same time the drivers but maintaining the user level library. In parallel new industrial PCI to HIPPI and PCI to Serial-HIPPI interfaces will be evaluated as they become available.

THE PCI TO HIPPI INTERFACE

The HIPPI to PCI modules are built around a PCI interface with a DMA Engine, a Scatter Gather Memory, a History Memory, and the HIPPI Interfaces. The block diagram for the Destination is shown in Fig 5. The only differences for the Source are that the data flow goes in the opposite direction and the use of a different HIPPI circuit.

As CERN developments are often very specialized, quantities for economic use of ASICs are rarely reached. For this reason the PCI interface has to be implemented using

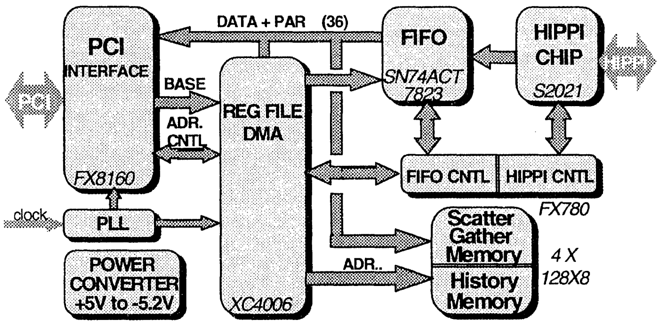


Fig 5: Block diagram of the PCI to HIPPI

programmable logic. The only FPGA for the PCI interface, available at the start of the project, that is fully compliant with the PCI AC-drive parameters was the iFX8160. This FPGA is complemented by a Xilinx XP 4006 containing register files, the DMA engine, and the memory control. As the iFX8160 has no possibilities to bring a buffered clock to the output pins an external Phased Lock Loop is used for clock distribution.

Interrupts

All interrupts except "End of Transfer" will be masked during transfer. To allow a simpler way of event building by using the connectivity control of the switch, an external interrupt on the back panel will be included and can replace the "End of Transfer" with an external signal coming from the system, such as "End of Spill".

The Scatter Gather Memory

In order to allow the sustained speed for receiving very large blocks of data, the host processor should not interact in the transfer process. In the HIPPI destination and in the HIPPI source, this is done by including a Scatter/Gather Memory and a History Memory. For every page boundary a scatter/gather index addresses the scatter gather memory and a new address is loaded into the DMA write pointer. This is the first address of the new page to be accessed. From here the DMA write pointer increments for every word transferred. At the end of the page a new page address is fetched the same way from the scatter/gather memory. The size of the scatter/gather memory is system determined, and depends on the number of pages required to store the maximum allowed quantity of data. At present 64 KB is foreseen which corresponds to a 256 MB transfer block on an Alpha DS 5000/200, and 512 MB on the new PCI stations. Bit 31 indicates the last page allowed for the current transfer. The scatter/gather memory must be initialized by the processor before start of the transfer, using I/O operations.

The History Memory

For the same reasons of speed, the pointers and messages concerning the transfer are not forwarded to the host, but stored in a history memory. Tags contain such data as "Burst Status", "End of Packet", "End of Connection" and LLRC and Parity errors. The offset to the beginning of the transfer is stored with the tag. The processor can access the history memory only after the transfer is finished, using normal I/O operations.

Scatter/gather memory and history memory fit together in a fast 4 MB (4 x 128x8) static memory that is included in the normal PCI address space of the Source and Destination boards as follows:

Base Address	Address Window	Unit
XXX	00000-3FFFF	Registers
XXX	40000-7FFFF	FIFO
XXX	80000-BFFFF	Scatter/Gather
XXX	C0000-FFFFF	History Memory

FIFO Memories and HIPPI Interfaces

Due to the 10 Km link to drive, large FIFOs, in the order of 4 Kwords (2 X 74ACT3651) are necessary. The HIPPI interfaces are built around AMCC circuits, the S2020 for the source and the S2021 for the Destination. A power converter generates the necessary voltages for HIPPI.

Solving the Mechanical problems

Mounting two 100-pin HIPPI connectors for a full duplex interface needs a double-width back

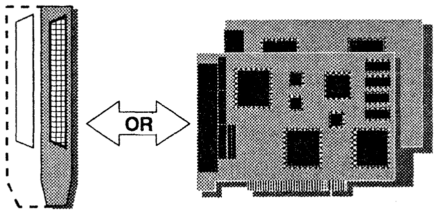


Fig 6: A double width panel vs. two modules

panel, or a special solution. Using an I/O cable into a remote connection box is too delicate in a HEP experiment environment. The logical solution is two independent units, one as a HIPPI Source and the other as a HIPPI Destination, as shown in Fig 6. At the same time the power dissipation per module is within the limits of the PCI specifications.

Alpha Specific Properties

Looking in depth at the present Alpha station, its internal circuitry uses the 21072-AA IC. An interesting point is that this processor bus to PCI interface chip, according to the datasheet, shows asymmetrical throughput for different speeds and modes:

Mode	Read/Write	Speed MB/s	HIPPI Function
DMA	W	120	Destination
DMA	R	70	Source
Pr. I/O	W	84	Destination
Pr. I/O	R	22	Source

To obtain speeds in the 70 - 80 MB/s range, the best results can be obtained if the HIPPI Destination uses DMA mode. However, programmed I/O seems to be fast enough. The HIPPI Source will have much better throughput using DMA transfer. During DMA transfers the processor should be idle; if programmed I/O is used the processor should only do the transfer and no other tasks.

Project Status

A prototype board has been mounted and is ready for tests. The rather simple implementation needs wait-states during set-up and page switching. This means that speeds will probably not be better than 60-70 MB/s, and as such not fast enough for the final needs of NA48. Having this solution working, the project will be continued with the development of 64-bit PCI interfaces where the 100 MB/s speed of HIPPI can be sustained. In this second version both Source and Destination modules will have both programmed I/O and DMA transfer possibilities.

CONCLUSION

Data acquisition for HEP has always been looking for more computing power with faster interconnect possibilities. In the past this was done with dedicated electronics and interfaces. The PCI gives the physics community an I/O standard that is fast enough to solve a large number of its speed problems. In addition, HEP experiments have a lifetime from design to end of operation of 20 years. This spans several generations of computer technology, therefore the interconnections have to be as flexible as possible. PCI and PMC offer a fast, processor independent, industrially supported solution.
PCI could be used on many places in future HEP data acquisition systems and its influence will certainly go much further than the few examples described here.

REFERENCES

PCI Local Bus Specification, Rev2.0, April 1993
PCI Special Interest Group, Oregon

Proposed Standard for a Common Mezzanine Card Family: CMC, IEEE P1386/Draft 1.5, September 1994

Proposed Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC, P1386.1/Draft 1.5, September 1994

Atlas Technical Proposal, Chapter "Trigger, DAQ and Computing",
WWW: http://atlasinfo.cern.ch:80/Atlas/Welcome.html

T. Anguelov, “HIPPI to TURBOchannel Interface” CERN/EAST note 93-07, 21 June 1993.

A. Van Praag, et al, “HIPPI Developments for CERN Experiments”, CERN/ECP 91-28, 7 November 1991. Presented at IEEE NSS 1991

Datasheet RIO2 8060/8061, Creative Electronic Systems, Geneva, 1994

SCI Subsystems for HEP Experiments, R.E.Hughes-Jones, et, al, Manchester University. October 1994. Presented Open Bus 1994, Paris

J-P. Matheys, et al, “Data Transfer and Distribution at 70 Mbytes/s”, CERN/ECP 93-7, 19 July 1993. Presented at IEEE RT 1993

CERN High Speed Interconnect project,
WWW: http://www.cern.ch/HSI

PC Architectures for Video Capture

Aki Kaniel
Marketing Manager, Digital Video Products
Philips Semiconductors , Sunnyvale, CA.

Abstract:

Video capture to PCI computers will be discussed. The real-time requirements and the solutions of video capture via CPU memory and direct to Graphics Frame Buffer will be discussed as well as the Philips chipset designed for these applications.

1. The Video Capture Challenges

Video capture has real time and high data rate requirements. Besides the raw bus bandwidth it is much more important to assure the availability of enough bandwidth at the right time. It is the nature of live multimedia data streams that the flow of video data can not be stopped. Input data, that is not captured, is lost. A multimedia presentation which is paused or has gaps and interruptions is quite disturbing. Scheduling and bandwidth allocation in a predictable manner on the video bus is a major issue. Dedicating a secondary bus for multimedia use reduces that problem, but it is still effected by the randomness of the main system access to the secondary bus.

A further criteria to measure the robustness of a bus is its capability to recover from an error and to handle conflicting demands beyond the safety margin of abundant bus bandwidth . The priority control for the arbiter on PCI can manage and distinguish the real-time importance of each master requesting the bus.

If the multimedia bus is so overloaded, that certain data can not be transmitted in time, this data will be lost. The source based and destination targeting DMA control of PCI transfers can easily recover from denied bus access, as the next data burst transfer of the affected data channel incorporates the actual physical address. The visual degradation of live video information by losing a couple of pixels is marginal, as long as the real time phase, i.e. position of transmitted pixels is maintained.

2. Video Capture subsystems

Video capture subsystems are found on add- on boards or on the mother board of personal computers. The main purpose of video capture subsystems, is to make the video information available for display on the computer screen, for example as video in a window, surrounded or overlaid by graphics. Add-on cards for PCI based PCs are based on two architectures. In both architectures analog video input is sampled, digitized, decoded into RGB or YUV signals and scaled.

In a Video Display Only system, the video data is stored in a frame buffer shared with the graphics engine. Note that the graphic controller need to have an Image Port to receive the video data stream.

In a Video Processing system, the video data is transferred via under DMA control to any memory in the computer (CPU memory, Hard Disk, communication CODEC, graphic display frame buffer). In both systems the merging of video and graphics signals is done in the digital domain.

Both architectures has significant advantage in size, cost and performance, as it eliminates the video controller, local image memory, dual ported RAMDACs, etc. Note that the inexpensive Video Display Only architecture does not make the video data available for other purposes than display, such as video compression or encoding to a VCR. The Video Processing architecture maintains system and application flexibility at a somewhat higher cost.

3. PCI Local Bus

Intel developed the Peripheral Components Interconnect (PCI) as a high bandwidth bus for next generation high performance desktop computer and workstation. PCI is a 32 (or 64) bit wide bus, with 17 or more auxiliary control signals, and a maximum clock frequency of 33 MHz. Addresses and data are multiplexed and travel on the same wires of the bus. The raw 32 bit bus data bandwidth is 132 MBytes/s. Multiple 'master' agents can be on the bus and access is dynamically granted by à central arbiter under control of a priority scheme and latency counters, which are set up by the operating system. Any client with master capability on the PCI has its own DMA control and can read from and write to any memory address in the system, without involvement of the CPU.

The PCI-SIG (Special Interest Group) supplemented the PCI design guide with a 'PCI for multimedia design guide'. It discusses considerations and gives advice for system configuration on the PCI bus supporting multimedia applications, without changing the actual specification of the bus. The major issues are how to organize the system in order to guarantee maximum latency for real-time signals, and to define the required amount of FIFO-buffering in particular devices. In planar systems where the CPU and the peripheral components reside on the same bus, the PCI bus can handle a certain amount of real-time multimedia data transfers under 'business' performance requirements and some reasonable assumption about the behavior of the rest of the system. For more demanding, more complex and high performance multimedia applications. e.g. with multiple live video pictures on the screen, it is recommended to utilize a secondary PCI bus, which is dedicated to the multimedia data transfers. PCI multimedia bus and system bus are connected via a 'bridge' containing FIFO buffers. The latency considerations regarding the multimedia PCI are much more predictable in that case.

4. System Solutions and Integrated Circuits

Philips Semiconductors offers various video Tuners and ICs for desktop video applications. The SAA7110 and SAA7111 are ideal for Video Display Only systems.

Several chipsets are available for Video processing systems. The SAA7116 is a Digital Video to PCI Interface IC. The SAA7116 incorporates a FIFO decoupling the real time video data stream from the PCI bus and provides two DMA channels to deliver the data in packed rasterized format for local display or planar format for compression applications. The SAA7116 has a glueless interface to other Philips video capture and scaling ICs like the SAA7110 and SAA7111 digital video decoders, SAA7186 and SAA7140 digital video scalar or SAA7196 digital video decoder and scaler.

Sound, Graphics, and MPEG Video on a Single PCI Card

Michael K. Harris & Tony Chu
Avance Logic, Inc.
47509 Seabridge Drive
Fremont, CA 94538
Ph. (510) 226-8555 Fax (510) 226-8039

ABSTRACT

The Peripheral Component Interconnect (PCI) bus is ideally suited for transferring large amounts of multi-media data to the new generation of audio and video products currently available from Avance Logic and other companies. One inherent drawback of the PCI bus has always been the "one-load" requirement which is why multi-function cards have not existed in the market. The other drawback of using the PCI bus for multi-media data transfers is the danger of overloading the system bus which is also used for graphics, mass-storage, network and other essential systems functions. The authors suggest a single device which integrates PCI to PCI and PCI to ISA bridges along with video input and processing functions. With this and other devices (PCI graphics, ISA sound, video/MPEG decoder), an all-in-one multimedia PCI card can be constructed. Furthermore, most multi-media data is transferred on the secondary PCI bus without occupying significant bandwidth of the primary PCI bus.

Performance and Backplane Positioning of PCI Adapter Cards

Dennis Aldridge
Director of Product Marketing
Texas Microsystems
P.O. Box 42963
Houston, TX 77242
(713) 541-8200/8226 (fax)

This session will present issues in use and performance of PCI adapter cards in front and behind the bridge.

CMC Mechanical Implementation

David C. Moore
Senior Hardware Engineer
Digital Equipment Corp.
129 Parker Street, PK02/J60
Maynard, MA 01754-2571
Ph. (508) 493-2257 Fax (508) 493-4659

BIOGRAPHY

David Moore, presently employed by Digital Equipment Corporation, for the past 17 years, and located in Maynard, MA. My current job title is Senior Hardware Engineer and my job function has been design and development of new computer products and to provide assistance in the development of open industrial standards. The specific standards that I have contributed to are IEEE 1301, 1301.1, 1301.2, 1301.4, 1156.1, 1156.2, 896.2, 1101.10 and am presently draft technical editor of 1386, 1386.1 and 1386.2.

ABSTRACT

The purpose of this paper is to provide the Common Mezzanine Card (CMC) designer with a more complete understanding of the design variations and how to use them to provide the necessary design features for their product. Much flexibility has been added to the CMC to provide the designer with flexibility to optimize his design. Along with this flexibility it is necessary to add constraints that may not be completely obvious at first glance. It is my intent to call attention to these areas and make them more easily understood. The variations which will be discussed are CMC sizes, CMC envelopes, CMC staking height/connector height/standoff height/bezel height vs. component height limits on CMC and host modules, optional voltage keying, number of mezzanine connectors, host module component placement and height restriction area designations

CMC SIZES

Dependent on real estate needed for the design, a CMC size can vary from a single wide standard depth to a double wide, extended depth. In all, there are four sizes available to the user. The basic sizes available are: 1) the single wide standard depth (75.0 mm X 150.0 mm), 2) the double wide standard depth (150.0 mm X 150.0 mm), 3) the single wide extended depth (75.0 mm X 250.0 mm), 4) the double wide extended depth (150.0 mm x 250.0 mm). These dimension depict the boundary limits of the CMC rather than actual card size. Since some systems cannot provide sufficient room for extended sized CMC the single wide standard depth is the preferred module size. Due to design requirements, more real estate or possible rear panel I/O connections, the wider or extended versions may be necessary.

CMC ENVELOPE

The total three dimensional envelope for a CMC module is the width and depth of the CMC (note four options listed above) plus a total component height of 8.2 mm with exceptions along the CMC bezel I/O area the standoff /keying area and the mezzanine connector positions identified in the CMC specification. The bezel I/O area provides for connector clearance up to 32.0 mm from the CMC bezel and an envelope height of 13.5 mm along this strip. When optional CMC connectors are not used, components may be positioned in these areas so long as their height is no greater than dimension H_P. Dimension H_P is dependent on the CMC stacking height and is from 2.30 mm for 8 mm stacking height up to 5.30 mm for 11 mm stacking height. No components are to be positioned in the area of either 3V or 5V keying pins. Other than the optional CMC size there are no options defined for the CMC envelope except the position the CMC is placed within the envelope. This can be better understood as we proceed to discuss the stacking height options.

CMC STACKING HEIGHT

It must be remembered that the three dimensional CMC envelope is placed in a constant position above any specific host module. The host sets the envelope position and the CMC provides the positioning within that envelope. The stacking height between the host module and the CMC is a combination of the host envelope positioning and the CMC placement. Examples of this are seen when creating a CMC for a VME64 module where the stacking height is set at 10 mm and a Futurebus+ module with the stacking height at 13 mm. Both CMC modules use the same height bezel, standoffs and connectors but the Futurebus+ host module provides a 3 mm spacer to add to the standoff and bezel height and also the host mounted connector must be increased by the 3 mm. The CMC is allowed to vary its position within the envelope to allow the designer freedom of component placement.

CMC COMPONENT HEIGHT

The component height on the CMC is only dependent on the position that the CMC resides within the CMC envelope. As the CMC moves closer to the host module, within its envelope, more component height is made available on CMC side 2 and less on side 1. Total component height for a CMC is 8.2 mm minus the CMC board thickness except in the area reserved for CMC front panel I/O. Maximum component height is limited to 0.2 mm less than the stacking height. These maximum component heights must be reduced by another 0.7 mm when components are not electrically isolated (conductive surfaces).

CMC MEZZANINE CONNECTOR

The CMC connector used (IEA E700 AAAB) is not optional but since the position that the CMC resides in is, it is necessary to comment. The host connector does not vary due to the CMC position within the CMC envelope so the CMC connector must make up for the variation. The assigned connector that corresponds to the stacking height shall be used.

HOST KEYING VOLTAGE

The host determines the signaling voltage. Since both 3.3V and 5V may be used it is necessary to prevent incompatible modules from connected. A keying pin is to be installed on the host to prevent this occurrence. Note a CMC that can operate on either voltage would provide clearance for both keying pin positions.

HOST MODULE COMPONENT PLACEMENT

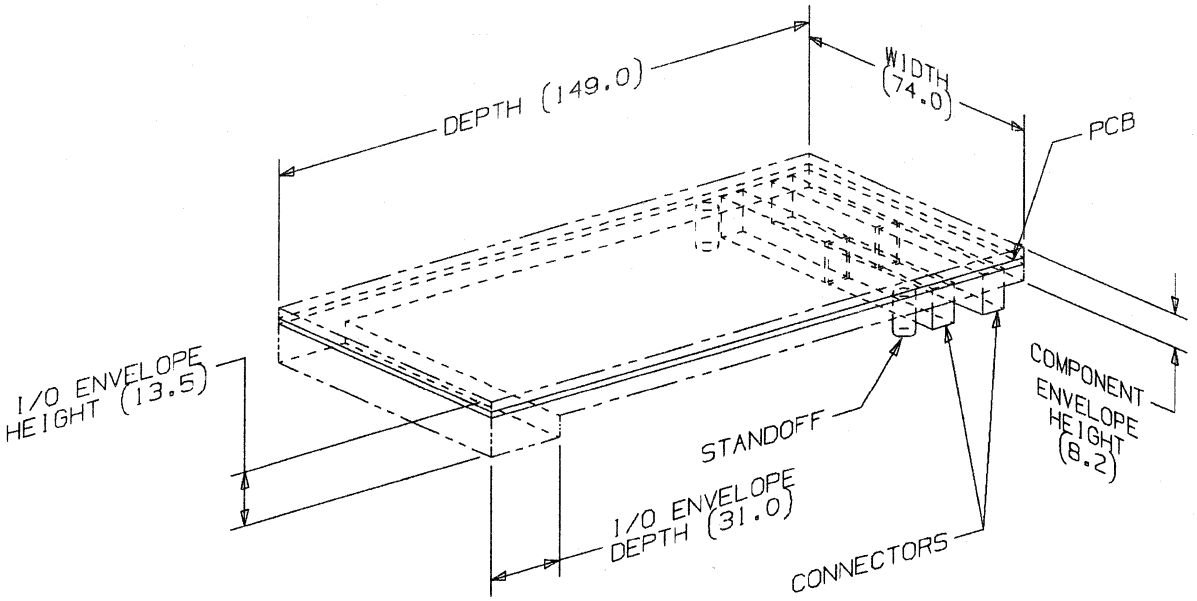
The main requirement for the host module is that it must provide the mechanical and electrical components to insure fit and function for all intended CMC options. Component placement on the host starts 32.0 mm from the back surface of the front panel and extends back in the area of the CMC to either 149.0 for standard or 251.0 mm for extended depth.

HOST COMPONENT HEIGHT LIMITS

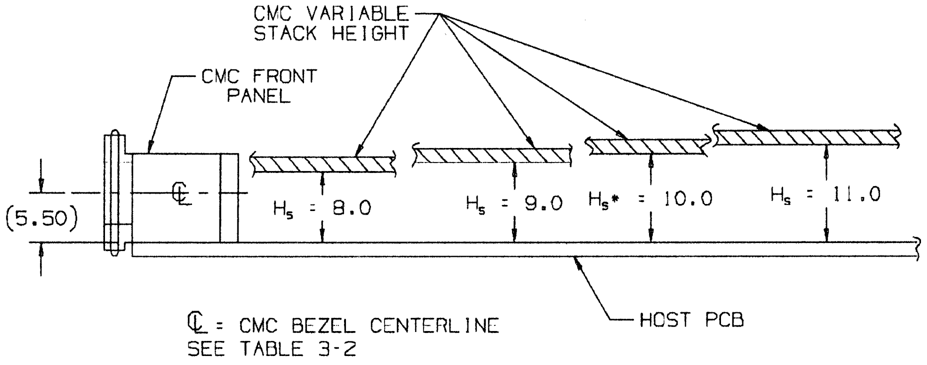
As stated earlier, maximum component spacing on the CMC in the I/O area is 9.8 mm. Note that no components shall be positioned in the I/O keepout area until the stacking height increases to greater than 10.0 mm . For host modules such as the Futurebus+ where the stacking height is 13 mm there is a allowable component height of 2.0 mm available indicated as restricted I/O area..

When a host designer builds a module which does not require all available mezzanine connectors he may choose to use the space that these connector are assigned for other components. This space may be used for components but it is necessary to limit the height of these components so that they would not interfere with a CMC that has all connector positions utilized.

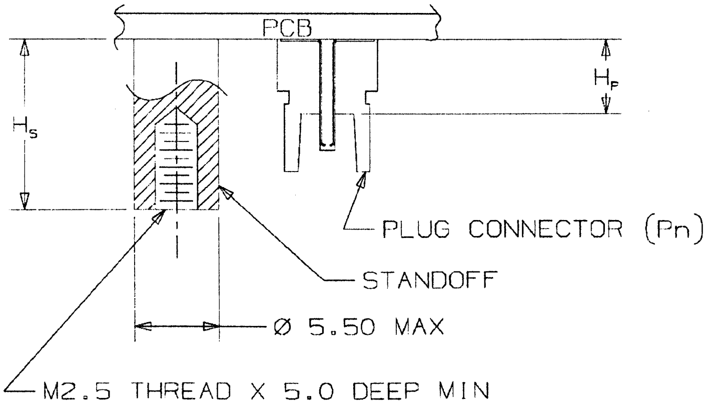
The component height limit for a 10.0 mm stacking height is 4.7 mm. The component height for the host module may increase 1 to 1 as the stacking height increase. I Futurebus+ at a stacking height of 13.0 mm may have components as high as 7.7 mm.


DEPTH (149.0)
WIDTH (74.0)
PCB
I/O ENVELOPE HEIGHT (13.5)
COMPONENT ENVELOPE HEIGHT (8.2)
STANDOFF
I/O ENVELOPE DEPTH (31.0)
CONNECTORS

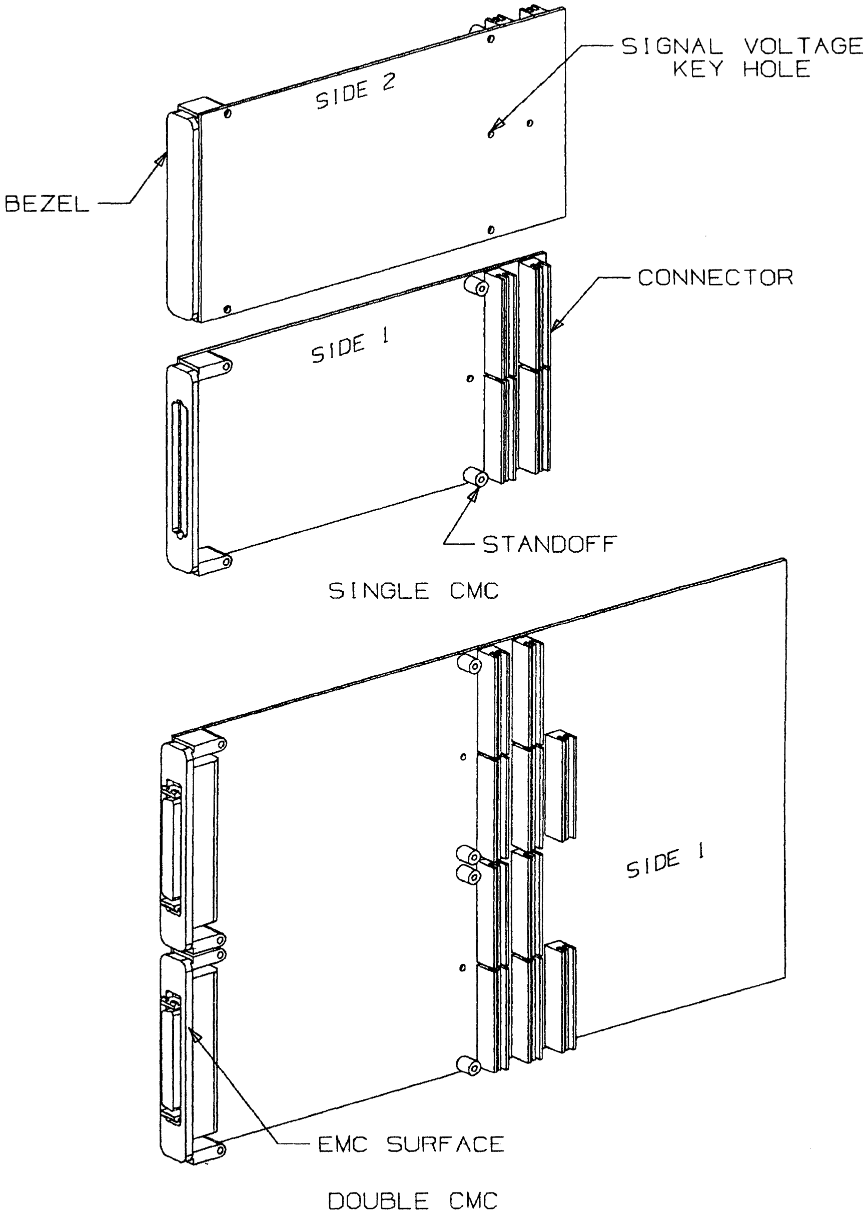
CMC Envelope



Mezzanine Stacking Height



HP = Maximum Component Placement in connector area



**Typical Single
and
Extended CMCs**

CARDBUS PC CARDS:
A NEW OPPORTUNITY

John Elmore
IBM Corporation, M/S-5432
1000 NW 51st Street,
Boca Raton, FL 33429-1328

The 16-bit PC Card interface provides performance capabilities equivalent to the ISA Bus. Since the majority of mobile systems now available are ISA based, this interface is ideal for I/O and memory expansion. The trend in the industry, however, is clearly toward 32-bit high performance systems with capabilities for addition of high performance expansion cards. Mobile systems must embrace the higher performance demands of the industry if they are to continue to enjoy their current robust rates of growth. Likewise, a means for attachment of high performance 32-bit expansion cards must be established to provide the versatility and convenience that mobile systems users now enjoy with 16-bit PC Cards. The CardBus PC Card interface is intended to enable this capability while maintaining support for existing 16-bit PC Cards. This paper briefly describes some of the opportunities presented by the CardBus PC Card interface

Characteristics

The CardBus PC Card interface enables many new PC Card applications and provides a means for the enhancement of current 16-bit PC Card product offerings. It introduces several important new capabilities and functions to PC Card applications, and is compatible with all new features and capabilities being introduced with the new PC Card Standard. CardBus PC Card features and capabilities include:

- 32-bits of Address and Data
- 33 MHz Operation
- Bus Master Operations
- Platform &O/S Independence
- Backward Compatibility
- interface Power Management
- 3.3 Volts (or lower) Operation
- Remote Wakeup
- Dynamic Reconfiguration
- Improved Audio Capability
- Non-customized Multi-function Card Support

The CardBus PC Card interface presents an opportunity to significantly expand the set of applications now available to PC Card users.

Opportunity

Clearly, the trend in the mobile industry is toward duplication of the office environment where increased performance and expanded functionality are the bywords to introduction of new products. Due to their cost, most high function and/or high performance features for office systems are under-the-covers add-in cards rather than standard system features. Interchange of these cards between platforms with disparate system buses is difficult if not impossible. The ability to use these features with a mobile system usually doesn't exist without the use of some type of 'docking station', which compromises mobility. The CardBus PC Card interface provides the opportunity for system and card manufacturers to provide a truly mobile high function/high performance capability for their customers, while maintaining the traditional PC Card advantage of interchangeability between systems, regardless of the type of bus employed by the system.

The CardBus PC Card interface provides improvements to 16-bit PC Card via a multiplexed 32-bit data/address interface which operates at up to 33 MHz (132 MB/s peak data throughput). Combined with bus master support, the CardBus PC Card interface enables system processor offload and optimization of multi-tasking operating system performance. Since CardBus PC Card sockets are also required to support 16-bit PC Cards, the motivation for implementation of the CardBus PC Card interface in all new 32-bit systems is strengthened. The implementation of CardBus PC Card interfaces in office systems will enable interchange of virtually any PC Card with mobile systems. Not only will it be easier to take more of your office functions with you, the opportunity for reduced capital expenditures is presented by the capability to use the same PC Cards in both environments. Further, customer satisfaction will be improved due to the cost savings and the improved configurability of their systems.

These are just a few of the more obvious opportunities and benefits presented by implementation of the CardBus PC Card interface. Without question, many more opportunities are immediately available and even more will emerge as technology and the industry progress.

PCI Synergy

The CardBus PC Card interface signaling protocol is derived from the Peripheral Component Interconnect (PCI) Local Bus signaling protocol. While there are some differences between the two specifications, operations are identical for most functions implemented. This similarity provides an opportunity for development of common silicon for use on both PCI cards and CardBus PC Cards. Refer to the PCI Mobile Design Guide and the new PC Card Guidelines documentation for further details.

Conclusion

The CardBus PC Card interface is the next-generation, high-performance 32-bit/bus master interface for PCMCIA/JEIDA. It provides the opportunity for migration of most high performance functions now available only on desk-top and larger systems to CardBus PC Cards for use in the mobile environment. New functions developed for CardBus PC Cards may also be used in 32-bit desk-top systems, if they are equipped with CardBus PC Card sockets.

While the CardBus PC Card interface is derived from PCI, it may be implemented on any 32-bit system that provides functionality similar to that provided by PCI. Although it is not the same as PCI in all respects, the signaling protocols are identical which enables development of common silicon.

All CardBus PC Card sockets must be able to accept and operate non-CardBus PC Cards within the capabilities of the system. Since all CardBus PC Card sockets also support non-CardBus PC Cards, the initiative for implementation lies with the system developers. Once CardBus PC Card sockets are available on 32-bit systems, CardBus PC Cards will be developed to take advantage of the performance provided by the 32-bit system and true office mobility will be realized.

INDUSTRIAL APPLICATIONS OF PCI

Jim Medeiros
Ziatech Corporation
1050 Southwood Drive
San Luis Obispo, CA 93401 USA

ABSTRACT

The Peripheral Component Interconnect (PCI) standard is the latest in a number of technologies originally created for "desktop" or "notebook" personal computers, and adopted by manufacturers of industrial computers. These technologies enhance computers used to automate control applications in the semiconductor, telecommunications, automotive, paper processing and other industries.

This presentation examines how PCI technology benefits industrial computers, and sorts through the evolving PCI choices available to designers of industrial control systems. PCI approaches discussed include passive backplane ISA-based computers (PICMG) the PCI Mezzanine Card (PMC) primarily for VME and MULTIBUS computers, PCMCIA Cardbus and Small Form Factor approaches, and rugged PCI formats for small format computers.

INDUSTRIAL APPLICATIONS OF PCI

The personal computer is traditionally associated with desktops at the office or at home, and increasingly in more transient, crowded quarters such as airplane seats or hotel rooms.

Yet the ongoing and rapid evolution of technology affecting personal computers reaches beyond the office and the aisle seat. Peripheral Component Interconnect (PCI), a high-performance local bus standard championed by Intel and now utilized by most major PC manufacturers, is just the latest in a long list of PC technologies borrowed by the makers of VME, MULTIBUS, passive backplane ISA, STD 32, and other industrial computer standards.

PC HELPS ADVANCE INDUSTRIAL COMPUTERS

It did not take industrial computer manufacturers long to figure out that aligning with the large PC market and taking advantage of that industry's advances in silicon and software would enable them to enhance their products in rapid-fire fashion. Industrial computer manufacturers also realized that the large volumes of the PC market would translate into quantities and cost savings that the smaller industrial computer market couldn't hope to beat.

Initially, it took industrial computer makers months and even years to adopt PC CPUs, operating systems, peripherals, video interfaces and networking capability, but today, there is little lag time between a technology's PC appearance and its industrial computer implementation.

Early industrial computers that were PC compatible often lagged the desktop marketplace by years with regard to performance and features.

In most cases a PC compatible embedded system would require multiple boards and many incompatibilities would exist because of the different environments.

In the last three years, the integration of PC compatible technology has dramatically affected the ability of industrial computer manufacturers to provide compact, fully PC compatible embedded computer systems. With this new technology (in some cases single chip solutions,) industrial computer systems can be every bit as powerful as those offered for the desktop. In fact, today it is possible to buy a Pentium based CPU, with hard disk, floppy disk, and Super VGA interface that will fit into the palm of your hand. *(See Figure A)*

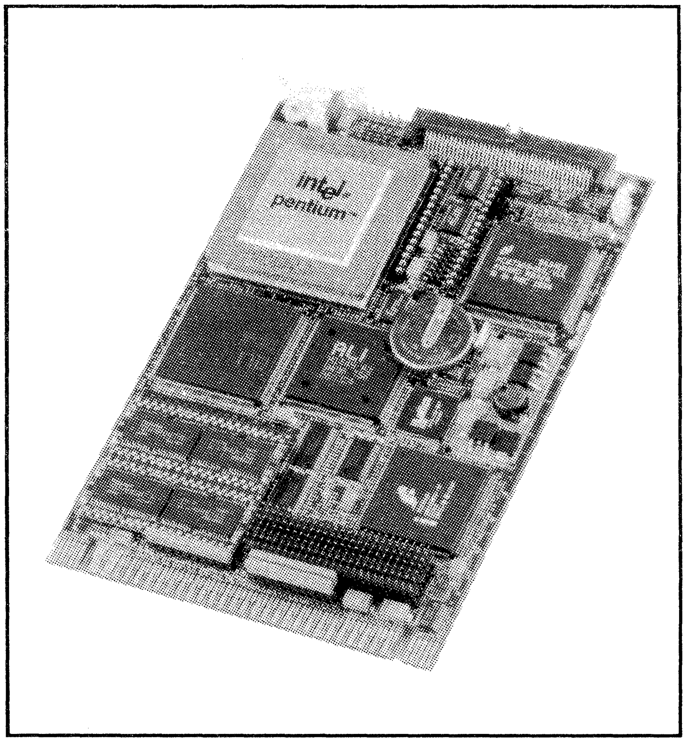


Figure A: A Compact Pentium Computer

Over the last ten years, industrial computer manufacturers have continued to improve the performance of embedded CPU systems by leveraging off of the latest in technology that is offered for desktop systems.

Prior to PCI, high speed local bus VGA interfaces have been available for embedded use. Early local bus architectures such as VL Bus have been primarily used only for VGA interfaces on 486 class machines. Although other interfaces have been made for VL Bus, the need for a standardized, processor-independent local bus has emerged to meet the performance requirements of PC compatible systems through the year 2000.

PCI: BREAKING DOWN THE BOTTLE NECK

Which brings us to PCI. This local bus solution to the bandwidth bottleneck created by increasingly demanding computer peripherals is rapidly invading not only desktop and notebook PCs, but industrial computers as well.

PCI products are relatively inexpensive and easy to design, and because the standard offers high performance and new sought after capabilities like "plug-n-play", PCI is becoming a prevalent part of industrial computers. Yet because the formats of these industrial computers differ quite considerably, the ways in which PCI is implemented varies widely. Emerging standards for these different PCI implementations will be discussed by industry representatives from Texas Microsystems, DEC, IBM, Intel and Ziatech Corporation.

Small PCI

Implementation and Strategy

Joseph F. DiMartino
Chairman -- Small PCI Workgroup

IBM PC Company
3039 Cornwallis Road
Research Triangle Park, NC 27709
(919) 543-9795
joe_dimartino@vnet.ibm.com

ABSTRACT

Peripheral Component Interface (PCI) is a bus standard that has gained industry wide acceptance in a relativity short amount of time. The robust characteristics of the bus have resulted in a variety of system designs leaning towards implementation. Some of the system designs are constrained by physical size, like mobile products, set top box's and small personal computers. The Small PCI (SPCI) form factor address's this design concern.

SPCI is an implementation of the standard PCI Bus in a physically smaller size. The electrical characteristics are similar and no additional silicon is required to implement this solution. The differences are in the connector header, 108 pins vs. the standard 124 pins as well as the card size which has the physical dimensions of current PCMCIA cards. Maintaining the aforementioned design characteristics result in relatively easy implementation of SPCI. Little invention is required because both PCI and the PCMCIA physical characteristics are well known and understood by the computer industry. Additionally the openness of the architecture insure multiple vendor sources and card manufacturer participation.

The proposed use of SPCI cards will initially address subsystems that are experiencing fast turns in performance, such as video, SCSI, audio, etc.. Also, expected to be implemented quickly are subsystems that have a high degree of user preference, such as communication cards (Token Ring vs. Ethernet, 10Base T, 10Base2 etc.). The above subsystems all have a major impact on inventory of adapter cards, system boards and finished goods. The implementation of SPCI will give system manufacturers leverage to customize systems at point of manufacture. This will reduce the risk of system board obsolescence and provide an opportunity to reduce the SKU's (stock keeping units) in inventory.

Session Description: D8B – PCI Compliance Forum

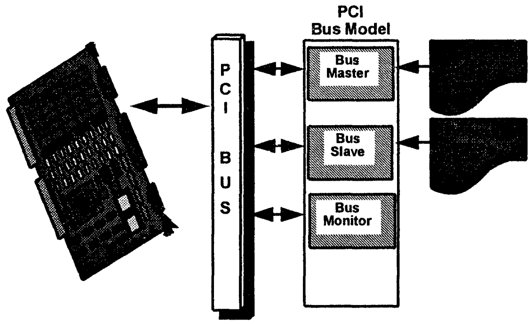
Conforming to a complex standard interface such as PCI requires that special consideration and attention be paid to issues surrounding compliance. The range of issues to be dealt with include mechanical, electrical, software, and hardware interface considerations. A number of methods are available for determining the level of compliance in a given design. This session will explore a number of alternatives which can work together to ensure compliance to the PCI specification and ease the burden of testing for compliance.

Using Simulation Models for PCI Compliance Verification

Dave Kresta
Product Manager
Logic Modeling Corp.
19500 NW Gibbs Drive
Beaverton, OR 97075
Ph. (503) 531-2249 Fax (503) 690-6906
E-Mail: een5sf@Leeds.ac.uk

Chair: Session D8B - PCI Compliance Forum

PCI Bus Model and Test Suite



SYNOPSYS®
LOGIC MODELING

1

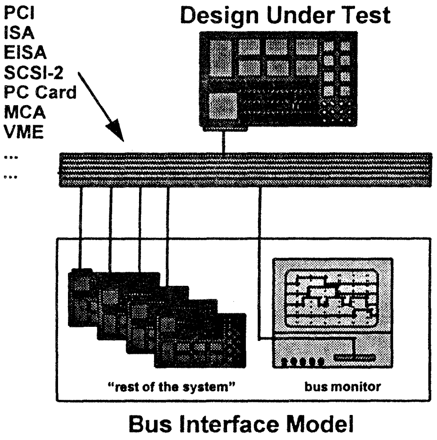
Agenda

- Bus Interface Models Overview
- PCI Bus Model
- PCI Test Suite
- PCI Design Kit

SYNOPSYS®
LOGIC MODELING

2

What is a Bus Interface Model?



Definition: *Simulation model allowing designers to verify a design's compliance to a particular specification, prior to prototyping:*

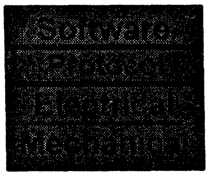
- **Emulates the behavior of the "rest of the system" - generates bus activity efficiently**
- **Monitors bus activity for protocol compliance.**

SYNOPSYS®
LOGIC MODELING

Bus Interface Models

SCOPE:

- **Bus Interface Models focus on hardware protocol level design issues.**



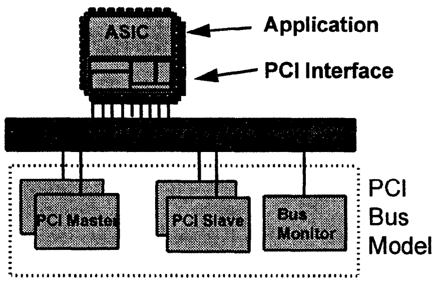
← Logic Modeling Bus Interface Models

SYNOPSYS®
LOGIC MODELING

Using Bus Interface Models

Primary Target: ASIC design

- Efficient test generation capabilities
- Pre-silicon compliance/interoperability testing

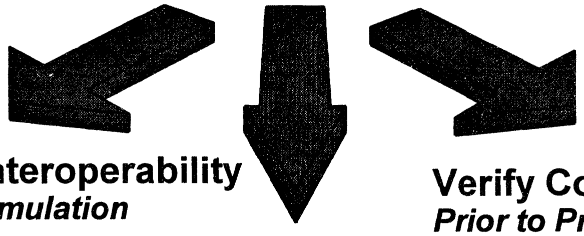


5

SYNOPSYS®
LOGIC MODELING

Bus Interface Models

Benefits



Test for Interoperability
During Simulation

Verify Compliance
Prior to Prototyping

Generate Test Vectors
More Efficiently

6

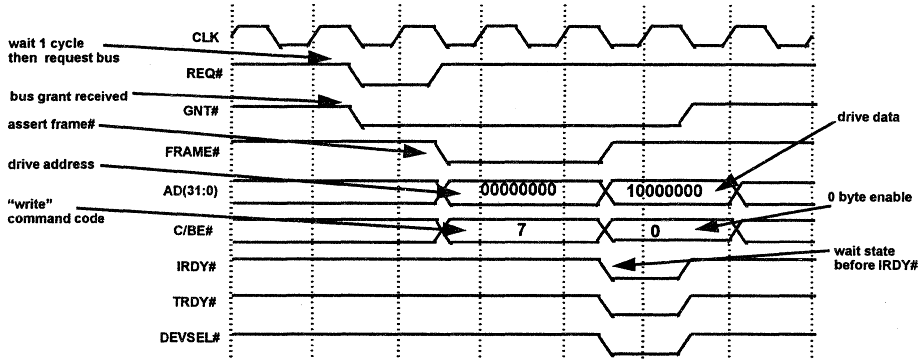
SYNOPSYS®
LOGIC MODELING

Automatic Protocol Generation

PCI Master model write command:

```
mem_write("00000000", 1, 0, 0, "10000000", 0, 0, false,false,false,linear);
```

Generated Bus Activity



SYNOPSYS®
LOGIC MODELING

7

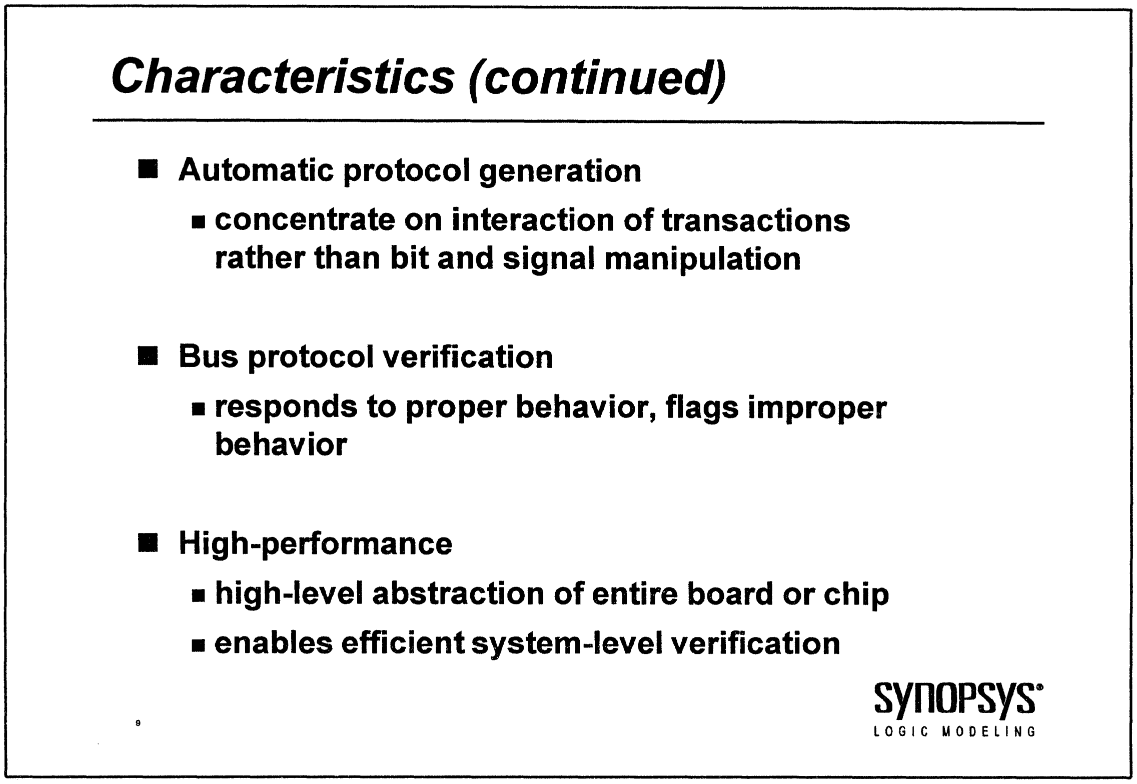
Characteristics of Bus Interface Models

- **Highly configurable, flexible**
  - **emulate any bus element: master, slave, etc**
  - **generate all bus cycle types**
- **Controlled via high-level commands**
  - **high productivity, high reuse**
  - **algorithmic control**
  - **efficient "test vector generator"**

```
read("....")
write("....)
get_pin("...")
```

SYNOPSYS®
LOGIC MODELING

8

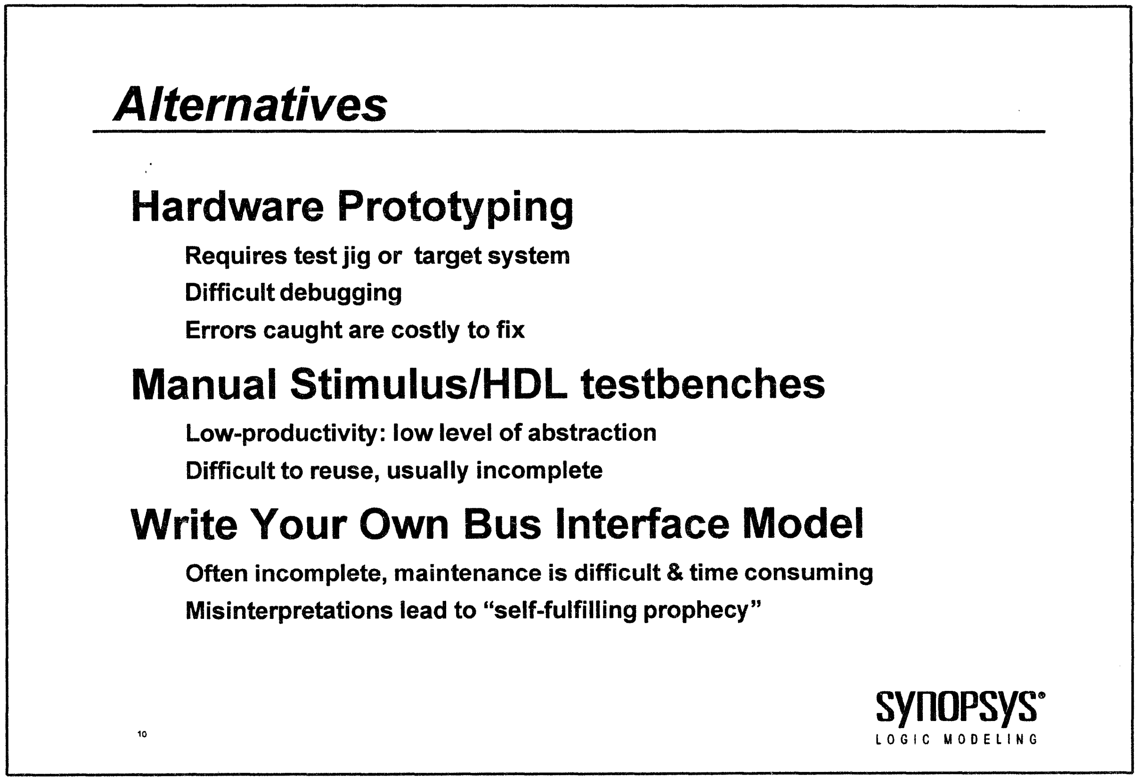


Characteristics (continued)

- Automatic protocol generation
  - concentrate on interaction of transactions rather than bit and signal manipulation
- Bus protocol verification
  - responds to proper behavior, flags improper behavior
- High-performance
  - high-level abstraction of entire board or chip
  - enables efficient system-level verification

9

SYNOPSYS®
LOGIC MODELING



Alternatives

Hardware Prototyping

Requires test jig or target system
Difficult debugging
Errors caught are costly to fix

Manual Stimulus/HDL testbenches

Low-productivity: low level of abstraction
Difficult to reuse, usually incomplete

Write Your Own Bus Interface Model

Often incomplete, maintenance is difficult & time consuming
Misinterpretations lead to “self-fulfilling prophecy”

10

SYNOPSYS®
LOGIC MODELING

Agenda Checkpoint

- Bus Interface Models Overview
- ➡ PCI Bus Model
- PCI Test Suite
- PCI Design Kit

11

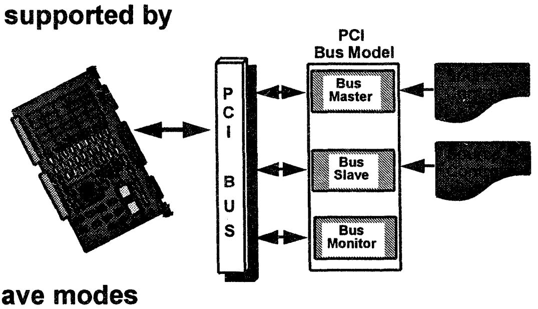
SYNOPSYS LOGIC MODELING

PCI Local Bus

Model of the PCI Local Bus - Specification 2.0

Initially developed by Intel in VHDL

Enhanced, distributed and supported by Logic Modeling



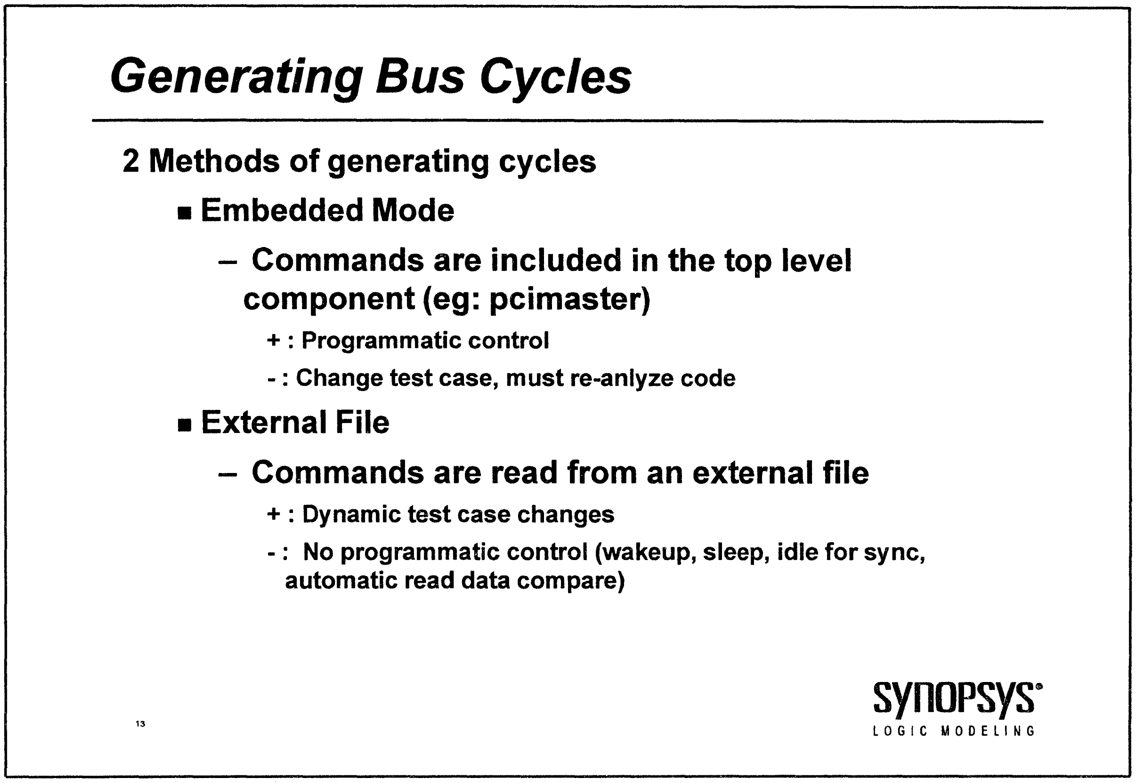
Models both master and slave modes

Model control is via user-written VHDL or Verilog

Distributed in VHDL and Verilog source code format

12

SYNOPSYS LOGIC MODELING



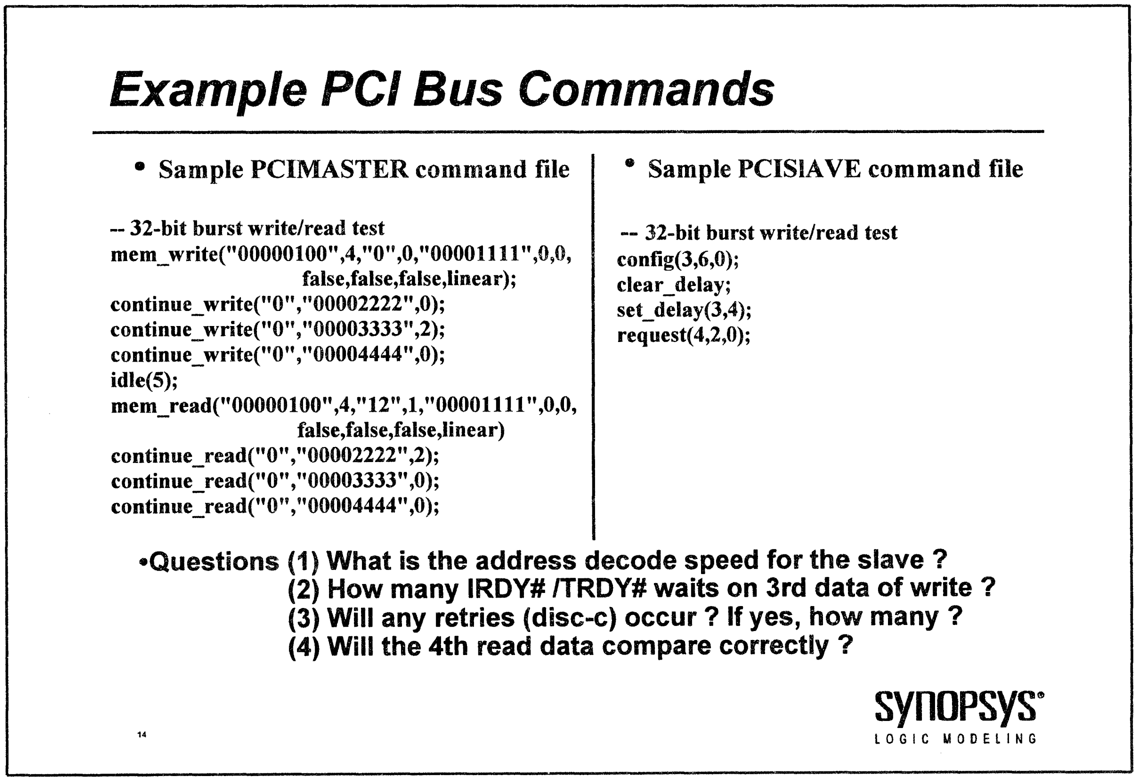
Generating Bus Cycles

2 Methods of generating cycles

- Embedded Mode
  - Commands are included in the top level component (eg: pcimaster)
    - + : Programmatic control
    - - : Change test case, must re-anlyze code
- External File
  - Commands are read from an external file
    - + : Dynamic test case changes
    - - : No programmatic control (wakeup, sleep, idle for sync, automatic read data compare)

13

SYNOPSYS®
LOGIC MODELING



Example PCI Bus Commands

- Sample PCIMASTER command file

```
-- 32-bit burst write/read test
mem_write("00000100",4,"0",0,"00001111",0,0,
                false,false,false,linear);
continue_write("0","00002222",0);
continue_write("0","00003333",2);
continue_write("0","00004444",0);
idle(5);
mem_read("00000100",4,"12",1,"00001111",0,0,
                false,false,false,linear)
continue_read("0","00002222",2);
continue_read("0","00003333",0);
continue_read("0","00004444",0);
```

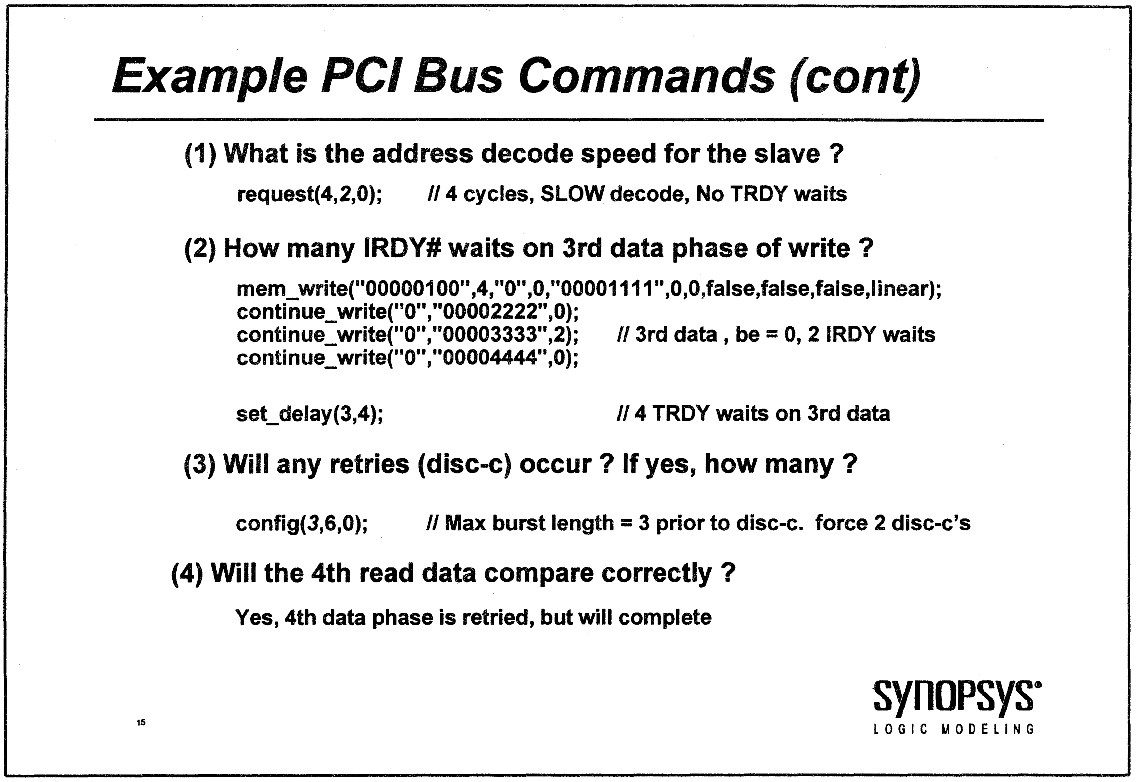
- Sample PCISlAVE command file

```
-- 32-bit burst write/read test
config(3,6,0);
clear_delay;
set_delay(3,4);
request(4,2,0);
```

•Questions (1) What is the address decode speed for the slave ?
(2) How many IRDY# /TRDY# waits on 3rd data of write ?
(3) Will any retries (disc-c) occur ? If yes, how many ?
(4) Will the 4th read data compare correctly ?

14

SYNOPSYS®
LOGIC MODELING



Example PCI Bus Commands (cont)

(1) What is the address decode speed for the slave ?

```
request(4,2,0);     // 4 cycles, SLOW decode, No TRDY waits
```

(2) How many IRDY# waits on 3rd data phase of write ?

```
mem_write("00000100",4,"0",0,"00001111",0,0,false,false,false,linear);
continue_write("0","00002222",0);
continue_write("0","00003333",2);     // 3rd data , be = 0, 2 IRDY waits
continue_write("0","00004444",0);

set_delay(3,4);                       // 4 TRDY waits on 3rd data
```

(3) Will any retries (disc-c) occur ? If yes, how many ?

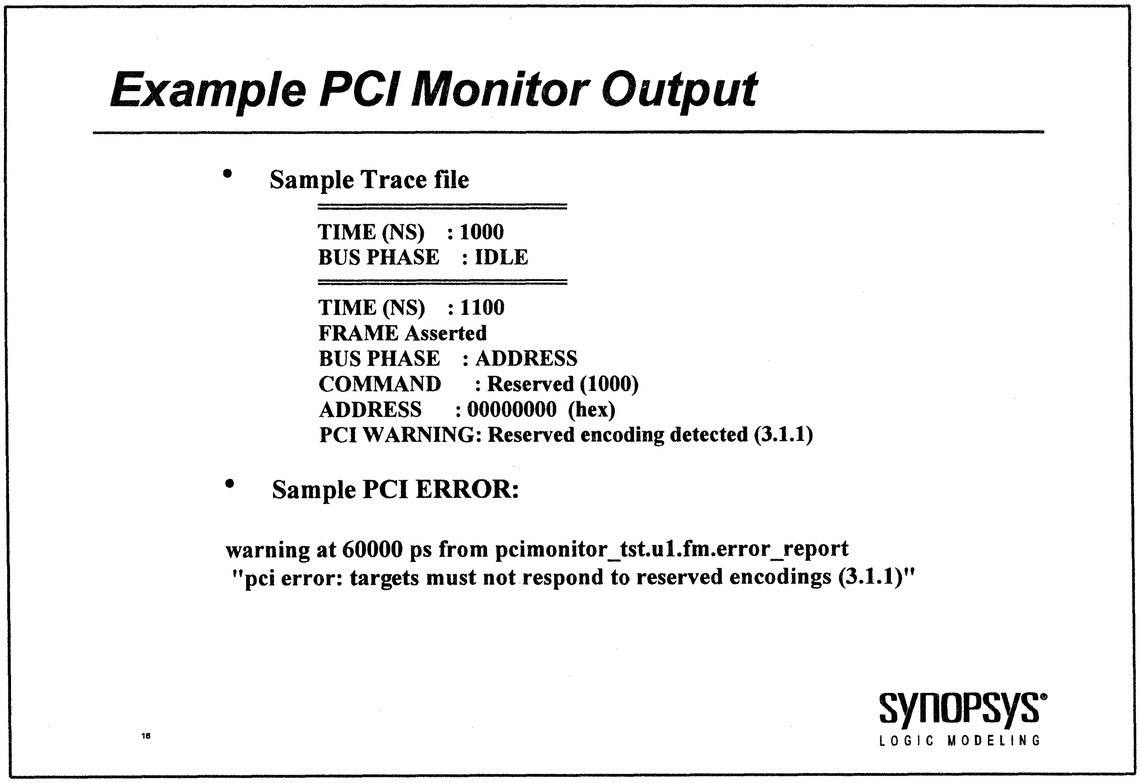
```
config(3,6,0);      // Max burst length = 3 prior to disc-c.  force 2 disc-c's
```

(4) Will the 4th read data compare correctly ?

Yes, 4th data phase is retried, but will complete

15

SYNOPSYS®
LOGIC MODELING



Example PCI Monitor Output

- Sample Trace file

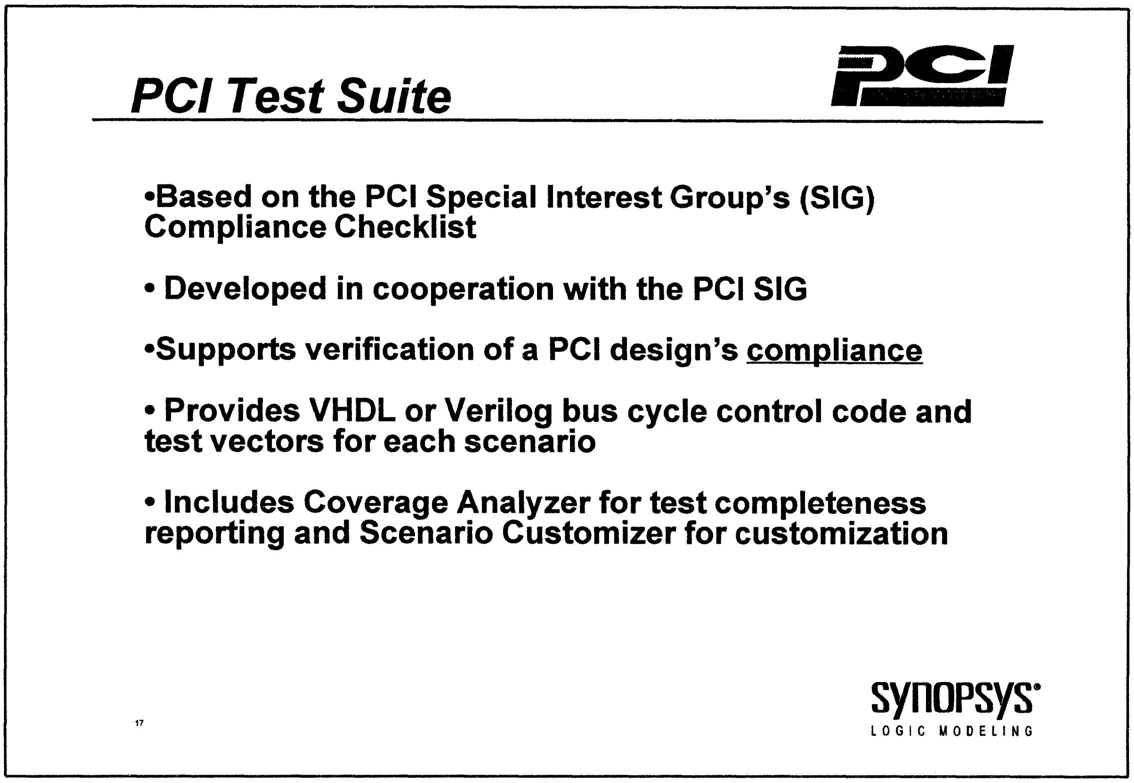
```
========================
TIME (NS)   : 1000
BUS PHASE   : IDLE
========================
TIME (NS)   : 1100
FRAME Asserted
BUS PHASE   : ADDRESS
COMMAND     : Reserved (1000)
ADDRESS     : 00000000 (hex)
PCI WARNING: Reserved encoding detected (3.1.1)
```

- Sample PCI ERROR:

```
warning at 60000 ps from pcimonitor_tst.u1.fm.error_report
 "pci error: targets must not respond to reserved encodings (3.1.1)"
```

16

SYNOPSYS®
LOGIC MODELING

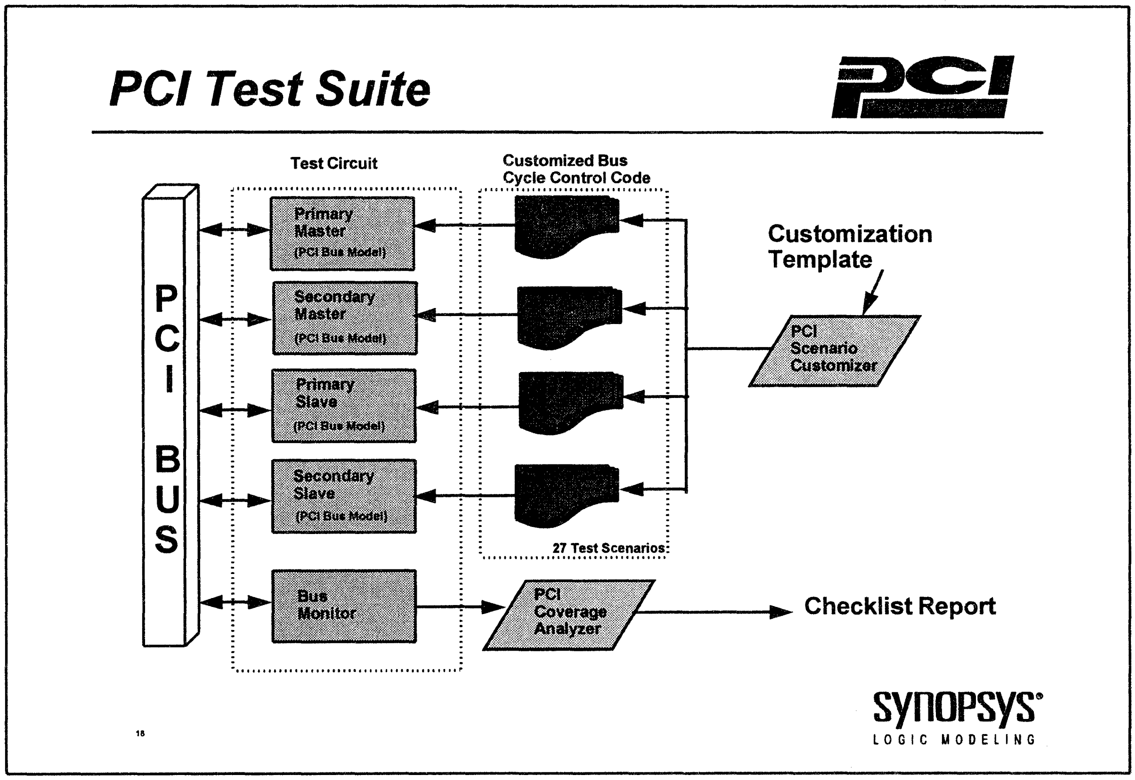


PCI Test Suite

- Based on the PCI Special Interest Group's (SIG) Compliance Checklist
- Developed in cooperation with the PCI SIG
- Supports verification of a PCI design's <u>compliance</u>
- Provides VHDL or Verilog bus cycle control code and test vectors for each scenario
- Includes Coverage Analyzer for test completeness reporting and Scenario Customizer for customization

17

SYNOPSYS®
LOGIC MODELING



PCI Test Suite

Test Circuit

Customized Bus Cycle Control Code

PCI BUS

Primary Master (PCI Bus Model)

Secondary Master (PCI Bus Model)

Primary Slave (PCI Bus Model)

Secondary Slave (PCI Bus Model)

Bus Monitor

27 Test Scenarios

Customization Template

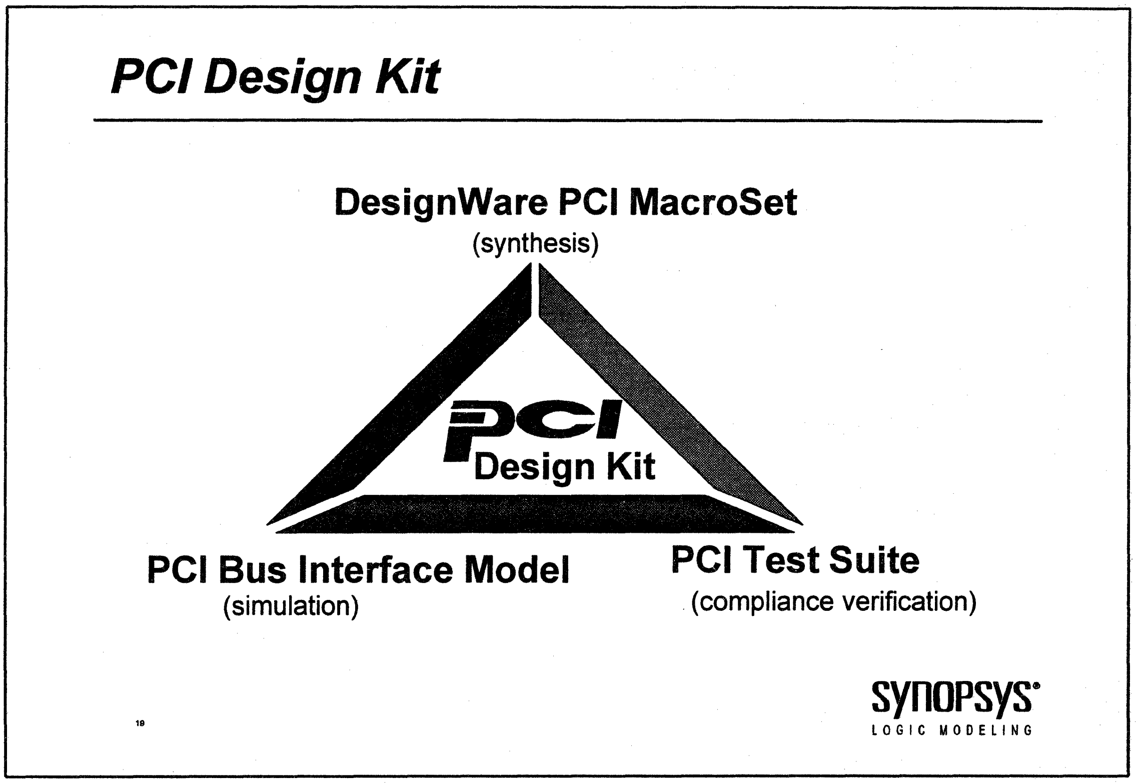
PCI Scenario Customizer

PCI Coverage Analyzer

Checklist Report

18

SYNOPSYS®
LOGIC MODELING



PCI Design Kit

DesignWare PCI MacroSet
(synthesis)

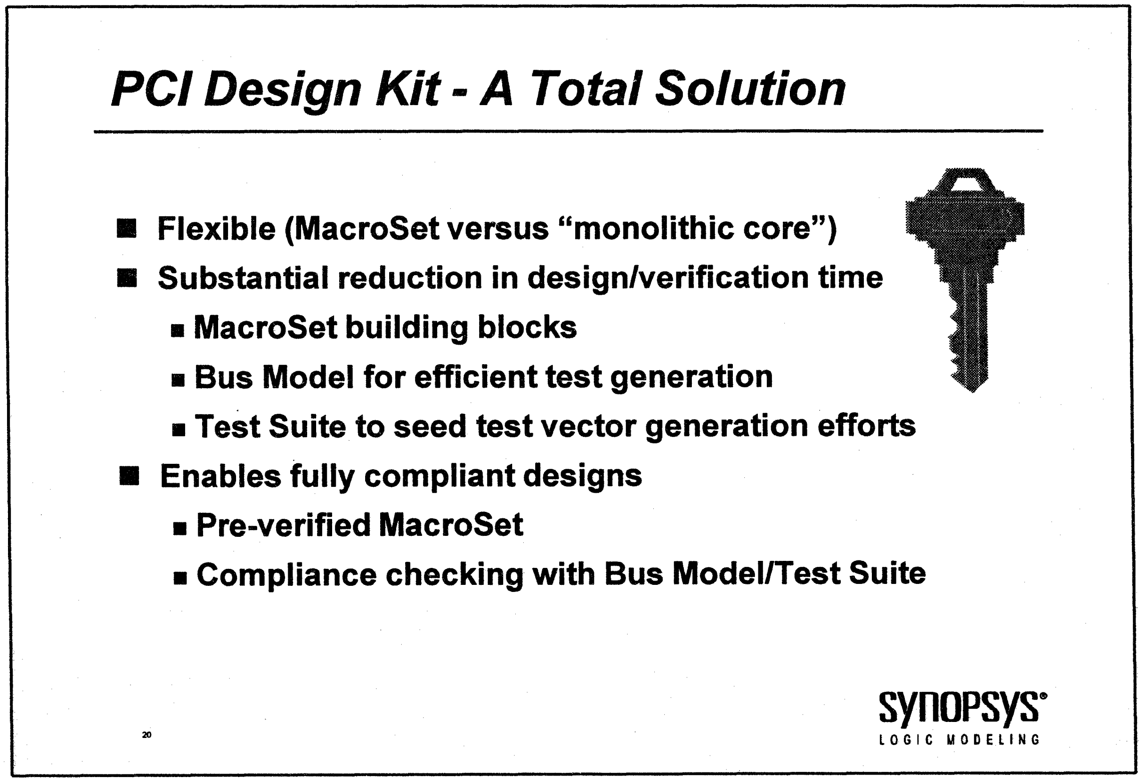
PCI
Design Kit

PCI Bus Interface Model
(simulation)

PCI Test Suite
(compliance verification)

19

SYNOPSYS®
LOGIC MODELING



PCI Design Kit - A Total Solution

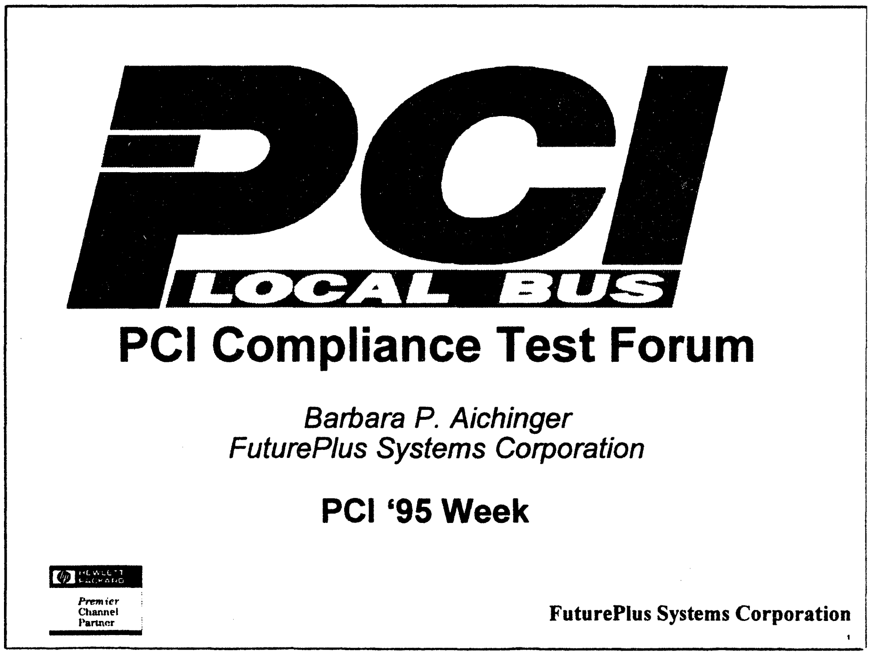
- Flexible (MacroSet versus "monolithic core")
- Substantial reduction in design/verification time
  - MacroSet building blocks
  - Bus Model for efficient test generation
  - Test Suite to seed test vector generation efforts
- Enables fully compliant designs
  - Pre-verified MacroSet
  - Compliance checking with Bus Model/Test Suite

20

SYNOPSYS®
LOGIC MODELING

PCI Compliance Testing

Barbara P. Aichinger
FuturePlus Systems Corporation
36 Olde English Rd.
Bedford, NH 03110
603-471-2734



PCI LOCAL BUS

PCI Compliance Test Forum

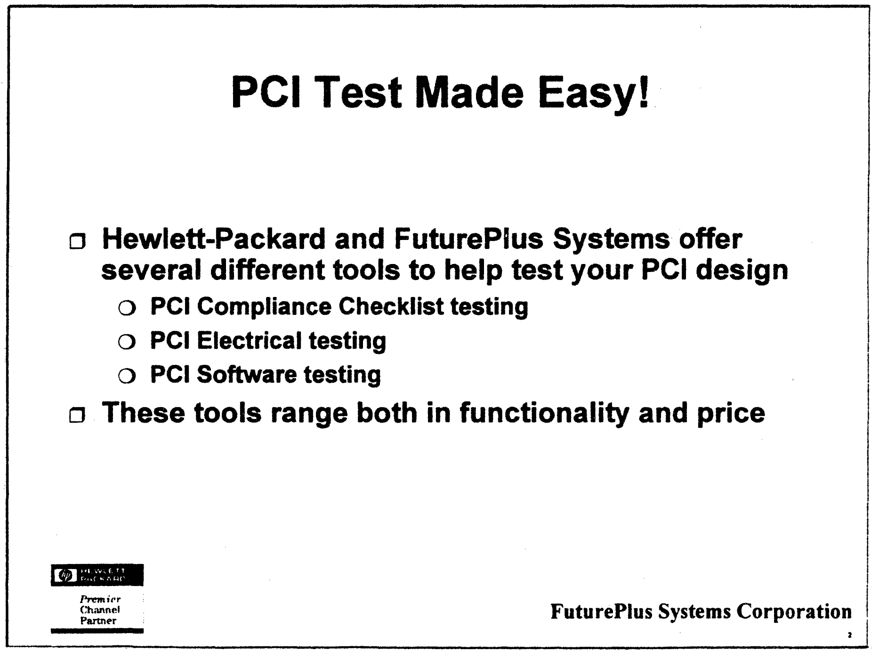
Barbara P. Aichinger
FuturePlus Systems Corporation

PCI '95 Week

HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

1



PCI Test Made Easy!

□ Hewlett-Packard and FuturePlus Systems offer several different tools to help test your PCI design
  - ○ PCI Compliance Checklist testing
  - ○ PCI Electrical testing
  - ○ PCI Software testing

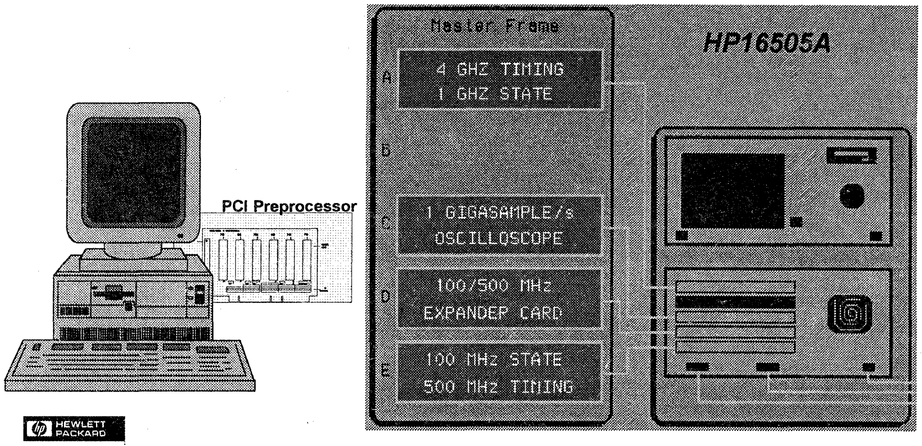
□ These tools range both in functionality and price

HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

2

Setting up your lab for PCI testing



HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

3

PCI Preprocessor and Extender Cards

- There are 3 cards available to help probe the PCI bus:
  - FS16P64 - 32/64 bit PCI Preprocessor
  - FS16P32E - 32 bit PCI Preprocessor and extender card
  - FSPS32 - 32 bit PCI Probe and extender card
- Summary of the differences:

Part Number	Extender card	HP connection	Test Points	Inverse Assembler
FS16P64	NO	YES	NO	YES
FS16P32E	YES	YES	YES	YES
FSPS32	YES	NO	YES	NO

HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

4

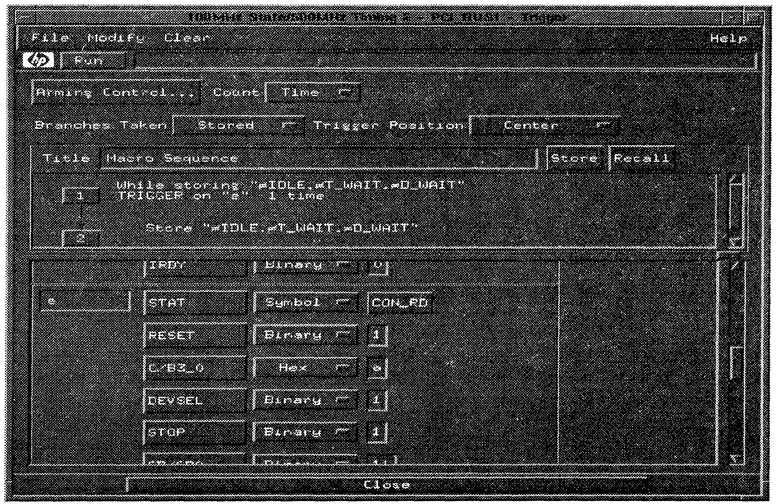
PCI Compliance Testing

- □ The logic analyzer, scope and high speed timing card can all work together
- □ Protocol, Configuration and BIOS Testing
  - ○ The trigger specification of the analyzer is very powerful....
    - ✓ Use the COMBINATION selection of the STORE specification to filter out.
      - ⇨IDLE states
      - ⇨Target initiated WAIT states
      - ⇨Master initiated WAIT states
    - ✓ Use the stored symbols to trigger on the event under test

HEWLETT PACKARD
Premier
Channel
Partner

FuturePlus Systems Corporation

Capturing PCI Bus Transactions



Count: Time

Branches Taken: Stored

Trigger Position: Center

Title: Macro Sequence

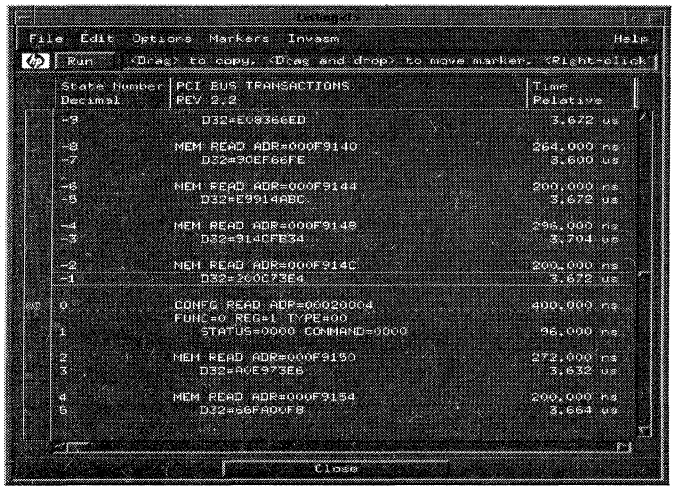
```
1  While storing "≠IDLE.≠T_WAIT.≠D_WAIT"
   TRIGGER on "a"  1 time

2     Store "≠IDLE.≠T_WAIT.≠D_WAIT"
```

	IRDY	Binary	0
a	STAT	Symbol	CON_RD
	RESET	Binary	1
	C/BE3_0	Hex	a
	DEVSEL	Binary	1
	STOP	Binary	1

HEWLETT PACKARD
Premier
Channel
Partner

FuturePlus Systems Corporation

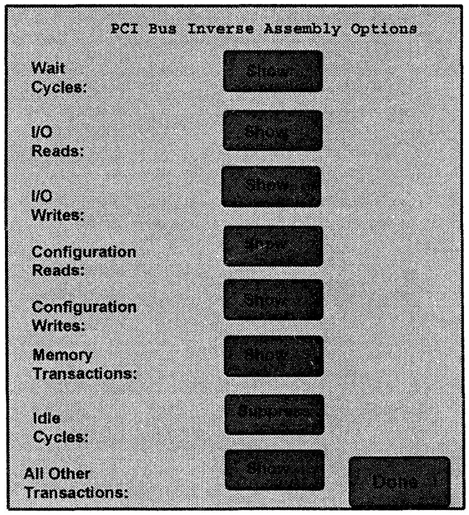
PCI Bus State Listing



```
<Drag> to copy. <Drag and drop> to move marker. <Right-click

State Number  PCI BUS TRANSACTIONS                            Time
Decimal       REV 2.2                                         Relative
 -9               D32=E0836 6ED                                 3.672 us

 -8           MEM READ ADR=000F9140                           264.000 ns
 -7               D32=90EF66FE                                  3.600 us

 -6           MEM READ ADR=000F9144                           200.000 ns
 -5               D32=E9914ABC                                  3.672 us

 -4           MEM READ ADR=000F9148                           296.000 ns
 -3               D32=914CFB34                                  3.704 us

 -2           MEM READ ADR=000F914C                           200.000 ns
 -1               D32=200C73E4                                  3.672 us

  0           CONFG READ ADR=00020004                         400.000 ns
              FUNC=0 REG=1 TYPE=00
  1               STATUS=0000 COMMAND=0000                     96.000 ns

  2           MEM READ ADR=000F9150                           272.000 ns
  3               D32=A0E973E6                                  3.632 us

  4           MEM READ ADR=000F9154                           200.000 ns
  5               D32=66FA00F8                                  3.664 us
```

Viewing PCI Bus Transactions

- ❑ Use the post processing display filters to filter the PCI bus transaction display.
  - ❍ Select to see
    - ✓ I/O Reads
    - ✓ I/O Writes
    - ✓ Configuration Reads
    - ✓ Configuration Writes
    - ✓ Memory transactions
    - ✓ No Idle states
    - ✓ No Wait states
    - ✓ All other transactions
  - ❍ OR ANY COMBINATION OF THE ABOVE!



PCI Bus Inverse Assembly Options

Wait Cycles: Show

I/O Reads: Show

I/O Writes: Show

Configuration Reads: Show

Configuration Writes: Show

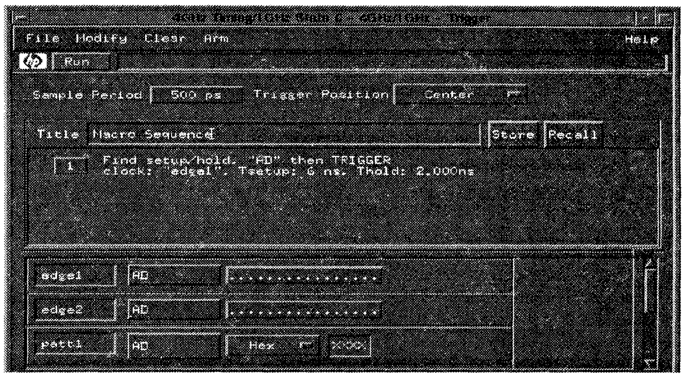
Memory Transactions: Show

Idle Cycles: Suppress

All Other Transactions: Show

PCI Compliance Testing

- Electrical Testing
  - Cross trigger the scope and the logic analyzer
  - Use the high speed timing card for ensuring that all setup and hold times are adhered to



Sample Period 500 ps Trigger Position Center

Title Macro Sequence

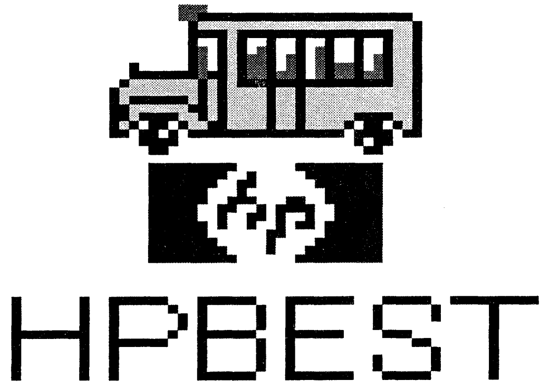
1 Find setup/hold. "AD" then TRIGGER
clock: "edge1". Tsetup: 6 ns. Thold: 2.000ns

edge1 AD ....

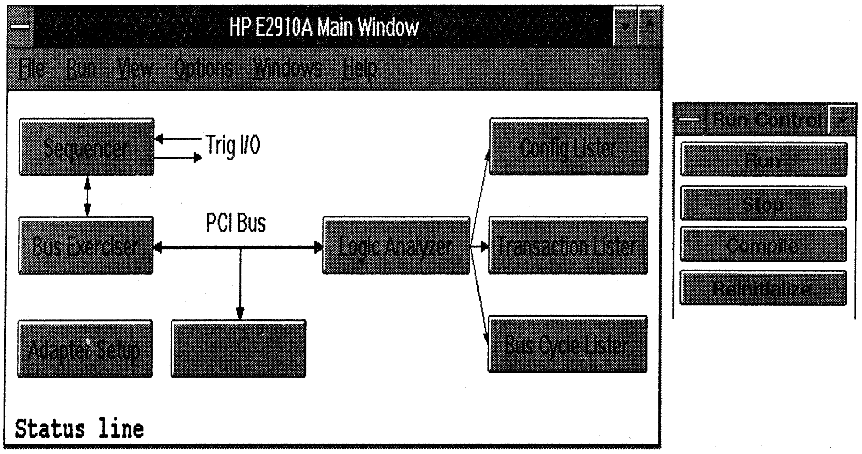
edge2 AD ....

patt1 AD Hex XXXX

HP BEST System



BEST Main Control Window

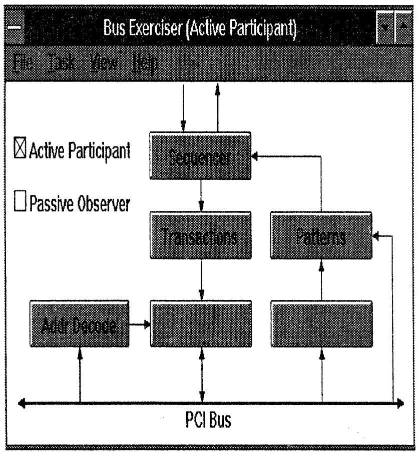


HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

Bus Exerciser Window, used to:

- ❒ Enter test sequences
- ❒ Define bus transactions using the transaction editor
- ❒ Define patterns
- ❒ Define address/transaction types to be decoded when acting as target
- ❒ Select whether passive monitor or active participant

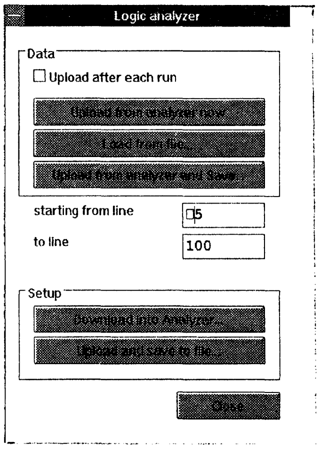


HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

Logic Analyzer Window, used to:

- ❑ Upload data from the Logic Analyzer for the Listers to process or save on disk
- ❑ Load previously uploaded data file from disk into the Data Listers
- ❑ Load the LA setup from file
- ❑ Define a range of data to be uploaded from the Logic Analyzer


Logic analyzer
Data
Upload from analyzer now
Load from file...
Upload from analyzer and Save...
Setup
Download into Analyzer...
Upload and save to file...
Close

□ Upload after each run

starting from line 05

to line 100

HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

13

Summary

- ❑ Time to Insight!...
  - ❍ If you can find it...you can fix it!
  - ❍ These tools are designed to shorten the *Time to Insight*
- ❑ It takes many different types of *testing* to bring a PCI product to market.
- ❑ In addition to Compliance testing, these tools are designed to help with system and general design verification testing.
- ❑ Come by the HP/FuturePlus Systems booth and take a test drive!

HEWLETT PACKARD
Premier Channel Partner

FuturePlus Systems Corporation

14

COMPLIANCE SEALS: VALUE OR BUST?

Lloyd Holder
NSTL
625 Ridge Pike
Conshohocken, PA 19428

I. Abstract

The practice of using compliance seals as a differentiator of products that meet certain test standards has worked well in those areas of IT technology that are dominated by a single vendor. Recent attempts at compliance testing programs with associated seals have not enjoyed the same success because competing vendors participating in committees have difficulty establishing and maintaining standards. It is time to remove the conflicts between marketing and engineering concerns in the interest of releasing more compliant and compatible products. It is more important for products to actually work together than for vendors to inform the market that products have negotiated a compliance obstacle course. The market itself has always been the final arbiter of all issues in the micro computer industry.

II. The Need for Compliance Seals

Given unlimited time and resources most engineering departments will be able to make most computer products work with each other. Unfortunately, market considerations usually contrive to limit the time and resources available to most engineering departments. This results in many computer products reaching market with limitations on their compatibility with other products.

New technologies must generally prove themselves to not only outperform existing technologies, but also to be as stable and as reliable. In their rush to gain a market edge some companies produce products that implement the new technology in an inconsistent manner. This in turn leads to confusion in the market since, in some cases, the new technology can be demonstrated as superior to the existing technology, while it may not work at all in other cases.

Engineers and marketing departments have traditionally sought ways to differentiate those products that exhibit good implementations of the new technology from the bad implementations. One popular method is the displaying of a compatibility or compliance seal. The general idea behind the seal is that the end user will perceive that all products exhibiting the seals will either work together or were at least tested together at some point in time. Large purchasers of IT products, when aware, will sometimes require that products qualify for these seals in order to participate in procurement bids.

Marketing departments generally demonstrate much more enthusiasm for compliance seals than the engineering departments. The desire to differentiate their products from the competition drives the marketing staff's enthusiasm. Obtaining the seals is their primary objective and they are not overly concerned about the methodologies used to determine which products earn the seals. Their secondary interest is in keeping the cost of obtaining the seal to a minimum, thereby retaining their competitive pricing and profit margins.

Engineering departments, on the other hand, are more concerned that the seals have true meaning to the end user. They are the ones who must defend and fix the products when problems occur. Their primary interest is in making sure that the testing methodologies are as extensive as practical. Most engineers feel that the more extensive the coverage of testing the greater the guarantee of compliance. Extending the testing methodologies is generally associated with increasing the costs for testing.

In practice, the nature and the amount of testing that actually occurs is a trade off between doing enough to placate the engineers without spending a sum that would result in a product selling price that is not competitive.

Compounding this situation are some other practical concerns. The current proliferation of seals has lead to a situation that could be referred to as "Seal Clutter". Product advertisements now contain so many compliance and award seals that it is doubtful that the end user still pays any attention to them. In the case of new technologies, like the PCI bus, standards must be established and it is not always clear who has the right to establish them. The successful compliance programs in the past have generally exhibited features that come from one of two cases.

III. The Dominant Vendor Case

The case of the IBM PC/DOS standard is a classic demonstration of standards being set by a vendor that dominates the market by capturing a significant portion of it. Prior to the introduction of the IBM PC, micro computers had flourished while exhibiting a variety of hardware busses and operating systems. Riding on the coattails of IBM's dominance in the mainframe and midrange markets, the IBM PC, a late entry into the micro market, eventually outsold all other competitors. Software vendors now had a stable market that consisted of a large installed base of common systems and they took advantage of this by writing software that was compatible with these systems. Adapter vendors also took advantage of this installed base by developing compatible products for its bus.

Because of its market dominance the IBM PC became the standard for the micro computer industry. Competitors felt compelled to demonstrate compliance with the IBM specifications even though they were not clearly published in detail. IBM has always demonstrated a reluctance to display any compatibility seals on any of its products. It further prohibits the issuing of any seals claiming IBM compatibility. In spite of this, and in the absence of seals, the term "IBM compatible" became an industry accepted concept.

IV. The Technology Owner Case

In the second case the market also plays a prominent role in establishing the standard. The most prominent example of this type of compatibility seal is the NetWare compatibility program run by Novell. By capturing a significant portion of the LAN market, Novell was able to put itself in a position where it could demand that hardware products demonstrate compatibility with its network operating system. As the owner of the technology Novell dictated the methodology that was used for testing compliance with its products. In the early days of development of the file server market, end users quickly adopted the NetWare seals as proof that the product was approved by Novell. The seals gave the purchaser the assurance of the technology owner that the products met the minimum requirements for compliance with the technology. Novell's original, relatively rigorous testing has since given way to a much more marketing oriented self-certification program granting products the NetWare Yes seal.

Other prominent industry owners of technology have followed the less rigorous self-certification testing idea. Included among the more visible ones are the Microsoft Windows and the Intel Inside seals. More recently, even IBM has modified its course with the introduction of OS/2 and Thinkpad seals. In some cases the use of the seal is primarily a marketing arrangement between the companies involved and it is often not necessary for the products to meet actual testing standards.

V. The PCI SIG Approach

The introduction of the ISA and Micro Channel busses by IBM guaranteed the simultaneous availability of compatible products. Both system and peripheral products were extensively tested together in IBM's labs before market release. Other busses in the micro computer industry have not enjoyed the same instant success in the compatibility arena. EISA, PCMCIA, VL-Bus, and then PCI all had their origins in an industry-wide specification rather than the proprietary technology of a single company marketing its products. By starting as public specifications these busses were saddled with initial products from many competing companies. Each company fiercely defended its unique interpretation of the specifications and the early products were not perceived by the end user as interchangeable.

It was only natural that the PCI SIG would eventually consider a compliance testing program that would yield products that display a PCI compliance seal. But with no dominant market player yet established a key ingredient for seal success was missing. At this point in the evolution of the IT market it is generally accepted that the successful adoption of new technologies is often associated with open specifications. Seeking industry-wide acceptance, the developers of the PCI specification were careful to avoid the perception that it was tied to a particular company or even chipset.

To compensate for the limitation imposed by the absence of a dominant player in the market, the PCI compliance testing program needed to establish a "golden suite" of products in order to get started. The mechanism for doing this was the "benchathon", a benchmark of interested parties for the purpose of establishing compatibility with each other's products. At this level of testing compliance with bus standards is compounded with other factors such as driver software and system resource conflicts. Unfortunately for the design engineers, it is only at this level that the market understands compatibility and compliance with standards. The initial results of the benchathons confirmed that the early PCI products were not nearly as interchangeable as they would need to be for general market adoption. These results were never published and the SIG has moved on to a lower level of compliance testing.

VI. Recent Trends

Advertisements for the Macintosh line of computers have been touting the Apple logo as the only true compatibility seal for some time now. Despite this trend the IBM/DOS compatible market still flourishes. This market has grown up accepting a certain amount of incompatibility as the cost of enjoying open standards and prices that are determined by the market rather than by the vendor community. This does not imply that vendors should reduce their compliance testing efforts. Rather, it should be accepted by the vendors that the market will always determine which products it considers compliant. The time is ripe for us to reduce our need for compliance seals and the difficulty associated with establishing a starting suite of accepted products from multiple vendors. It is much more important to establish a testing environment where the engineers from interested vendors can work freely towards compliance with specifications as well as on compatibility with each other's products. This will lead to much more interchangeable products and therefore quicker and greater market acceptance.

REAL-TIME ON-BOARD BUS TESTING

Jeffery A. Floyd
Matt Perry, Ph.D.
Motorola, Inc.
Semiconductor Product Sector, Austin, Texas

ABSTRACT

This paper will define and describe the need and methodology for testing wide buses in real-time and at speed. In today's environment, computer buses are growing along with system clock speeds. These wide high-speed buses require special attention at time of board layout and analysis. Characterization of or tests on a bus at high-speed cause an additional unnecessary problem of tester interference in the circuit. For example, the addition of bus loading and capacitance by a logic analyzer is detrimental to the circuit performance at high speeds. We have developed a technique to allow full-fault testing of these wide buses at multiple speeds, in real-time, without tester interference, using pseudo-random pattern generation, and allowing multiple seed and characteristic equations. The technique involves transmitting test patterns from multiple physical locations on the bus (one at a time) and simultaneously receiving the data at other points along the bus. This allows testing of the bus in a real-life environment and allows the test engineer to evaluate the design in adverse operating environments with little or no undue influence on the design. We accomplish this without tester interference, at high speed, and with little device pin-out overhead by using the IEEE JTAG protocol to control and access the test logic.

DEFINITION OF PROBLEM

In the past when an engineer wanted to test a new hardware design, he/she had to connect a logic analyzer or oscilloscope to the circuit. These test tools measure voltage over time that the engineer analyzes to determine if a circuit is functioning properly. This methodology is still usable for circuits of low speed and low density. Today, however, the speed of designs continues to rise with each new design generation; for example, some of the more popular personal computer designs are now approaching 150 MHz in CPU clock speed.

The density of the circuits is also increasing with each new generation. Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs) allow designers to take larger portions of their designs and place them on fewer ICs. The result is reduction in board size and in the spacing between the circuits, which increases the density of the circuit board.

It is difficult to test high pin count devices with small lead pitch with standard logic probes, and special sockets are expensive and add capacitance to the circuit. In high-speed designs, the test equipment (including probe, socket, and tester) significantly influences the circuit. All of the above factors combine to make the job of the engineer testing or debugging a design much more difficult.

Tester Loading

Any test device, from the most basic logic probe to the most exotic logic analyzer or the best automated tester, will impose its characteristics on the circuit it is testing. These characteristics present themselves as capacitance, inductance, and resistance; therefore, impedance changes in the circuit. Additionally, the tester will require that the driving device of the circuit being tested supplies more power to compensate for the additional load imposed by the tester. This can push marginally functional circuits into a non-functional state or cause a non-functional circuit to suddenly function.

Tester Influence at High Speed

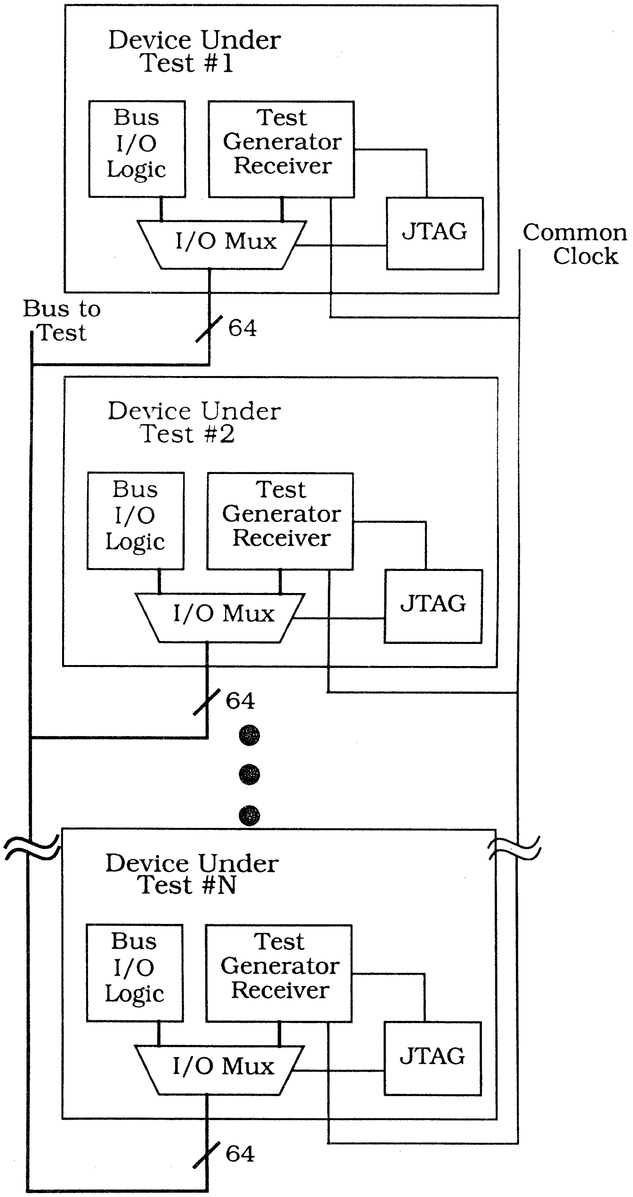
High-speed circuits have additional problems. The additions of the testers probe capacitance, inductance, and resistance to the circuit; changing the reflection dynamics of the circuit, that is, electrical reflections in the circuit now have an additional path to follow.

If a circuit is running in an environment with high EMI concentration or fails as the clock speeds approach maximum (for variable speed circuits), the introduction of the tester loading makes it impossible to accurately trace a problem to its root cause. The impact of the tester loading changes the behavior of the circuit and its performance.

On-Chip Testing Circuits

One obvious solution, and the topic of this paper, is to add the testing logic to the ASIC itself. This allows the test engineer to test and characterize a circuit without adding any unwanted electrical influence to the circuit. The test logic will, in most cases, add some propagation delay to the design. The design specification for the device should include this propagation delay compensation. Figure 1 shows the basic block diagram of the test logic.

Figure 1: System Block



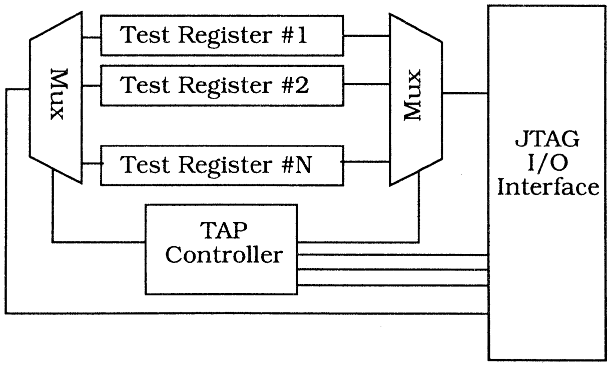
As shown in Figure 1, a JTAG control box controls the input/output multiplexor and a test generator/receiver block. The test generator/receiver generates a test pattern on one device and receives and verifies the pattern on all other devices on that same bus. This requires the I/O multiplexor configuration to be an output for the pattern generator device and an input on the pattern receivers. The bus I/O logic block is the block of logic that drives the bus during normal operations.

Figure 1 also shows a common clock. This clock could be the clock that operates the bus under normal conditions or a special test clock. Both have their appeal. In the case of the normal operation clock, you test the function and the skew of the real clock distribution as well as data transfer. If that clock line is damaged or incorrectly designed for high-speed transmission, then a secondary clock source is required. In this case, the use of the JTAG clock as a test clock provides a clock source and gives meaningful data. However, a 10 MHz limit imposed by the JTAG specification will not allow high-speed testing, but will allow a full set of connection tests to be performed. The addition clock sources into the system provides a greater range of testing possibilities.

JTAG

Figure 2 shows the contents of the JTAG block. For more detailed information consult the IEEE Joint Test Action Group (JTAG) 1149.1 specification.

Figure 2: JTAG Block



Briefly, the JTAG block is accessible via a defined five signal or pin port. The protocol running on these signals allows users to access a number of internal registers. If desired, the designer can allow access to a number of normal mode user registers and/or a number of test registers. In this example, we will assume that the JTAG controller has access to at least the following registers above the minimum requirements set forward in the JTAG specification:

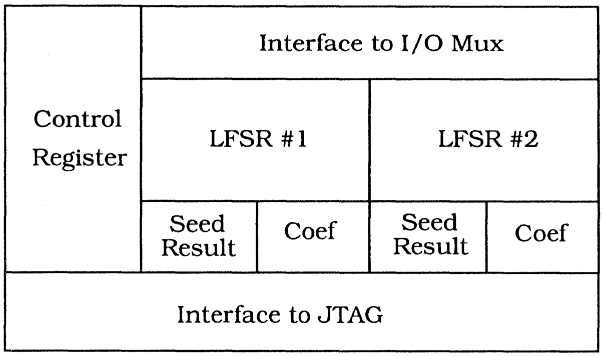
1. I/O Multiplexor Control Register
2. Test Generator/Receiver Seed Register
3. Test Generator/Receiver Result Register
4. Test Generator/Receiver Control Register
5. Test Generator/Receiver Coefficient Register

Via these registers, the test engineer can control and retrieve test information from the test logic described in this paper.

Test Generator/Receiver

The test generator/receiver block contains the circuits shown in Figure 3.

Figure 3: Test Generator/Receiver Block



The Linear Feedback Shift Registers (LFSRs) generate a pseudo-random test pattern at the test generator device, or receive and verify a test pattern at the receiver devices. This architecture shows the implementation of two LFSRs. The bus to be tested in Figure 1 is 64 bits wide. If each of the LFSRs in Figure 3 are designed to be 32 bits wide, two additional features are possible: one, the LFSR is easier to design and analyze, and two, the outputs of each LFSR can be interleaved on the 64-bit bus to allow for a more random and robust test of signals that are "neighbors" to each other on the 64-bit bus.

The seed/result registers write as seed registers and read as result registers. The seed register gives the LFSR a known starting point for the test pattern generation. The result register allows the engineer to retrieve the final result.

The coefficient registers allow the test engineer to change the characteristic equations of the LFSRs. These characteristic equations determine the complexity of the test pattern.

The control register starts and stops the LFSRs and controls the direction of data through the I/O multiplexor. It also selects the source of the test clock, normal operations clock, or JTAG or other secondary clock source.

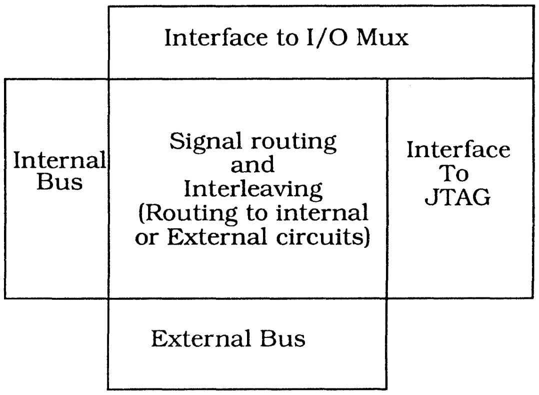
The interface to the JTAG block allows the JTAG Test Access Port (TAP) controller to read and write the control, seed, result, and coefficient registers. This interface controls the test process.

The interface to the I/O Multiplexor routes signals to the I/O Multiplexor.

I/O Multiplexor

The I/O Multiplexor contains the functional blocks shown in Figure 4.

Figure 4: I/O Multiplexor


Interface to I/O Mux
Internal Bus
Signal routing and Interleaving (Routing to internal or External circuits)
Interface To JTAG
External Bus

The I/O multiplexor routes and interleaves signals to the bus that requires testing. This block could also route the output of one or both of the LFSRs to internal circuits for internal Build In Self Test (BIST).

TESTING PROCEDURE

The following sections outline the process steps required to start the bus testing, what happens during the testing procedure, and how to retrieve and analyze the results. The steps outlined are for the general case; specific applications of this technology may require additional special process steps.

Starting Test

To start a test, the test engineer first selects which chip will be the test generator and which devices will be test receivers. This selection of transmitting and receiving devices creates a test "view". If the transmission device is in the middle of the physical bus and the receiving devices are on either end, this creates one view of the system. If, however, the transmission device is on one end of the physical bus and the receiving devices distributed along the bus, then this is a different view of the system. For the most complete and robust testing, all devices that are not the generator-configured device are receiver-configured.

The next step is to program all devices with the same coefficient and seeds. All LFSR #1 seeds and coefficients MUST be the same and the LFSR #2 seeds and coefficients MUST be the same. However, the settings for LFSR #1 may be different from the setting for LFSR #2. In fact, differences in the seed and coefficients for LFSR #1 and LFSR #2 creates a more robust testing environment.

The third step is to start the test. Writing to the chain of control registers via the JTAG protocol such that all control registers update on the same clock edge starts the test. Sufficient information written to the control register must start all LFSRs running and select a clock source. The same test clock source must be selected for all pattern generation and receiver devices.

During the Test

The test should begin with the generator device sending out pseudo-random patterns to the receiver devices over the bus to be tested. The receiving devices retrieve the pattern from the bus. One of the features of the LFSR is reducing the amount of test data required to find a specific fault.

Retrieving & Analyzing Results

When the test is complete, the receiving LFSRs should have the predicted pattern in them. If the pattern read from the result register is not as predicted or not all of the patterns match, then a fault detection occurred. There are several cases that represent the possible failure states that result by analysis of result registers. We discuss three below.

Case 1

All received patterns read as predicted and are consistent.

In this case, the test passed with no faults detected. For the most comprehensive results additional testing should be done with different coefficient register settings and different devices selected as the transmission device. This will change the test pattern and test view, possibly detecting faults that were undetectable with the previous pattern and view.

Case 2

All received patterns are the same but are not the predicted pattern.

In this case, a physical fault (a bus open, short, or stuck-at fault) detection most likely occurred. Further testing requires that the physical faults be corrected. It is possible to determine which line is at fault by analyzing the result pattern. Design simulation determines what a resulting pattern would be if a single fault was in the system. A reference to a table of possible results that stem from a single fault in the system identifies single faults. The complexity of this analysis grows exponentially when there are multiple faults in the system.

Case 3

The received patterns are different from the predicted ones.

In this case, there is the possibility of several faults. One possibility is an open, short, or stuck-at fault in addition to some transmission line effect problem. The best approach for this case is to reduce the clock speed and re-run the test. If an open, short, or stuck-at fault detection occurred, after correction of the fault, re-run the low-speed test to verify correct connections. Then the high-speed clock testing determines if a transmission line effect or some other high-speed problem exists.

If the low-speed tests reveal no problems and the results start to detect faults as the clock speed increases, then the most likely problem is a transmission line effect or EMI problem. Analysis of these results with a good set of board design analysis tools and/or a high quality network analyzer or time domain reflectometer allows the engineer to narrow the set of possible problems. With these tools the engineer can begin to trace a problem to the layout error or eliminate the possibility of transmission line effect and focus on the EMI issue. Testing in an EMI-shielded room verifies the presence of an EMI related issue. Careful testing to assure that the monitoring and testing equipment is not the source of EMI is important in this case.

Unfortunately, to correct a transmission line effect problem due to a layout error, the engineer will probably need to spin a new board. However, the data that results from the testing outlined in this paper will allow the engineer to re-design a board with a higher level of confidence in the new revision.

Regardless of the results, the engineer has the ability within this architecture to change the device on the bus that is acting as the transmitter of test patterns, changing the test view. When the test view changes, and tests are re-run with these new settings, additional data results. This information is useful for determining if one of the devices has a driver problem or if the faults are consistent, which points to a hard bus fault or transmission problem.

APPLICATIONS

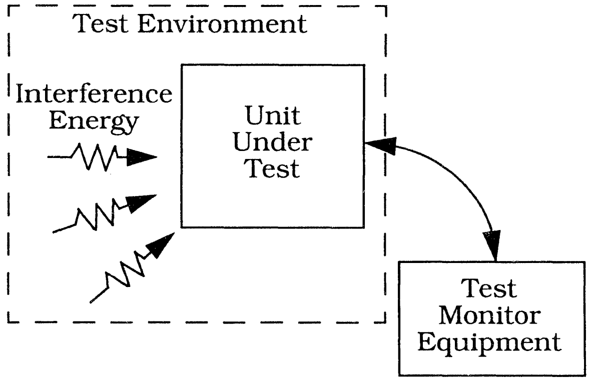
This section outlines some generalized applications of this technology. Additional applications are possible; the following examples are not intended to be exhaustive.

Adverse Condition Operation Testing

If a device design specifies a tolerance for a specific EMI level, the methods outlined in this paper allow a test engineer to verify functionality of the design in the presence of that EMI, the. The engineer verifies design tolerance by subjecting the design to an EMI source with the specified output level and running the tests under otherwise normal conditions.

Figure 5 shows interference energy directed at a unit under test. This energy takes many forms: heat, radiation, EMI, vibration, and or mechanical stress.

Figure 5: Adverse Environment Testing



On-Line Real-Time Testing

In a computer or other system that has a high reliability requirement and where undetected faults can linger for long periods of time before producing an error, a method of on-line system diagnostics is needed to periodically test the system. The method outlined here can be adapted to run tests on a system bus in real time during either an assigned test bus phase or idle data phases.

Figure 6 shows a possible distribution of test and data phases on a bus. These test phases take one or more forms:

1. Set number of bus clocks once a test phase is initiated.
2. Test phase automatically initiates when bus is idle.
3. Testing is done when a test task of low system priority is run.
4. Testing is done when a test task of low to medium priority (for systems with high reliability requirements) is run.
5. A complete test is done over several test phases to limit the impact of testing.

Figure 6: On-Line Real-Time Testing

Bus Phases over Time

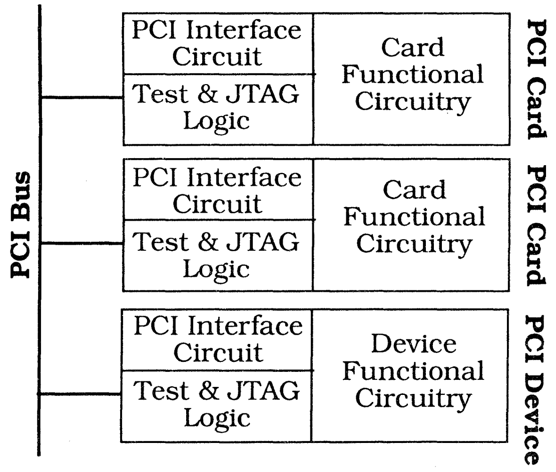
Data Phase	Test Phase	Data Phase	Data Phase	Test Phase

Time ———►

Specific Bus Application

In some modern computer buses the designers allowed for the addition of test interfaces in the specification. The PCI specification set aside five traces on the bus dedicated for the use of some testing mechanism. If these bus signals implement the JTAG protocol with a JTAG scan controller device on the mother board, then the methodology described in this paper can be used to implement testing of that bus. If each PCI device implements the test logic described here, then real-time bus testing during idle or dedicated testing phases generates on-line diagnostics information for fault analysis. If a computer board is thought to be faulty, then this same methodology generates test patterns and locates faults via a JTAG-aware board tester quickly and efficiently.

Figure 7: PCI Application Example



As shown in Figure 7, this testing methodology fits nicely into the standard PCI bus configuration, assuming the implementation of JTAG in the dedicated testing signals of the PCI bus. Incorporation of the test logic into both PCI card interface ICs and stand-alone PCI aware ICs and the addition of dedicated testing phases allows the test engineer to fully test a design quickly and accurately. The addition of logic to allow automatic or semi-automatic testing during bus idle time provides for a more reliable system.

CONCLUSIONS

This paper has demonstrated that it is possible to have on-device test logic that can generate meaningful test results of high-speed buses without unwanted tester loading. We have shown that this test logic can have multiple uses within the device by generating test patterns for not only a bus but also internal logic blocks. In addition, we have shown that this test logic is accessible by existing and standard methods via the JTAG protocol.

Application examples are provided to allow a potential user of this technology to design this functionality into new devices, providing their customers with a more reliable platform. Implementation of this technology also allows problems to be found more quickly with a high degree of accuracy for products with quick time-to-market requirements.

There are, as with any new design concept or idea, advantages and disadvantages:

Advantages:

1. LFSR can be used for other tests.
2. Nearly eliminates the need for test devices that load circuit, possibly past the point of normal operation.
3. Accessible via a standard JTAG protocol.
4. Testing from multiple "views" of the system.

Disadvantages

1. Requires additional logic to generate and receive the test patterns.

Further Research

There are areas that require more research and study for this technology. They are listed below:

1. Possible test results from common failure modes in a general or typical application. A set of failure modes exists that produces a common and generic result pattern for a given set of design configurations.
2. Set of LFSR characterization equations that produce the most comprehensive fault coverage in a given set of design configurations.
3. The possibility of testing in the absence of a common clock or on asynchronous buses.
4. The impact and or gain of on-line testing in real-time high-reliability systems using this methodology.

REFERENCES

The list of references below give the reader pointers to more detailed information on the design of LFSRs, EMI testing, and transmission line effects. These documents should be consulted at the selection of characteristic equations. They should also be consulted in the result analysis phase to help determine the reason for a given result.

[1] Abramovici, M., M.A. Breuer, and A.D. Friedman, *Digital Testing and Testable Design,* Computer Science Press, 1990.

[2] Bardell, Paul H. and William H. McAnney and Jacob Savir, "*Built-in Test for VLSI: Pseudorandom techniques*", Wiley, New York, 1987

[3] Chan, John C. "Boundary Walking Test: An Accelerated Scan Method For Greater System Reliability", IEEE Transaction on Reliability, Vol 41 No. 4 Dec 1992, pp. 496-503

[4] Daher, John K. and Joey M. Goodroe, "A Radiated Susceptibility Test Technique for PC Boards Implementing Built-In Test or Boundary Scan Designs." in Symposium Record of the IEEE 1990 International Symposium on Electromagnetic Compatibility. (Washington, D.C., Aug 21-23, 1990) pp.109-112.

[5] Divekar, Dileep and Raj Raghuram, and Paul K. A. Wang, "Simulate Transmission-Line Effects in High-Speed PCB and MCM Systems" Electronic Design, April 11, 1991, pp.113-122

[6] IEEE, 1990. Joint Test Action Group 1149.1 Specification.

DESIGNING HIGH PERFORMANCE PCI ADAPTERS AND EMBEDDED SYSTEMS

Mike Salameh
PLX Technology, Inc.
625 Clyde Avenue
Mountain View, CA 94043
415-694-2800

ABSTRACT

PCI (Peripheral Component Interconnect) bus enables performance and functionality improvement and cost reduction for computer adapters and embedded systems. Along with these advantages, however, comes increased interface logic complexity and gate count.

This paper describes several alternative architectures for PCI adapters and embedded systems and discusses their cost and performance tradeoffs. The paper explains methods for implementing these designs easily and inexpensively, focusing on the higher performance adapters and embedded systems.

In designing PCI adapters, engineers have a range of cost and performance alternatives ranging from simple slaves to intelligent masters with 32 bit RISC processors. In general, intelligent adapters perform their function (e.g. disk control, communications) faster and with less host CPU intervention than non-intelligent bus master or slave adapters.

By incorporating PCI as a "backplane" or I/O bus, embedded systems also benefit, and not just from PCI's high bandwidth. They also realize all the advantages of using a high-volume PC-standard architecture: wide availability of low-cost I/O silicon, a proven standard architecture and compatibility with other manufacturers' hardware.

PLX Technology's family of four PCI interface chips implements a broad range of adapter and embedded system architectures compactly and inexpensively. The design examples in this paper use PLX chips to implement high performance features.

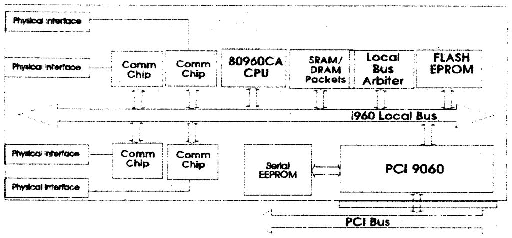
PCI ADAPTER DESIGN ALTERNATIVES

Two major alternatives for building a PCI adapter function are to use one of the many single chip PCI controllers available or to design a special adapter to meet specific performance and functionality requirements. There are many single chip PCI controllers available including graphics, SCSI, LAN and IDE. Clearly, if a single chip controller is available that meets the adapter's performance and functionality requirement, this is likely to be the lowest cost and easiest to implement solution.

For many applications, however, single chip PCI I/O controllers are not adequate. Intelligence, often in the form of an on-board 32 bit RISC processor, is required to meet the performance requirements. In some cases, the right single chip solution is simply not available. In these cases, the designer can choose from a variety of adapter architectures.

Intelligent Adapters

"Intelligent" adapters contain an on-board processor. The following example shows a multi-port communication adapter. XPoint Technologies, Inc. builds a family of such adapters. The communication chips could be 10 or 100 Megabit Ethernet, Token Ring, ATM, Fiber Channel or any other communications protocol. The processor in this example is an Intel 80960CA 32 bit RISC CPU. Also included are an on-board memory for data and instruction storage and a boot FLASH or ROM. The final piece is a chip, in this case PLX Technology's PCI 9060, to connect the adapter subsystem to the PCI bus.



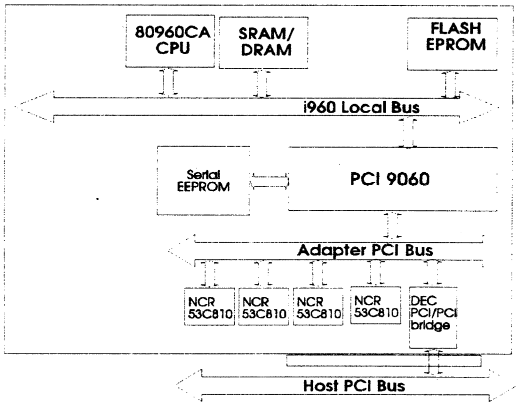
Multi-port Intelligent Communication Adapter

Performance Advantages of Intelligent Adapters The major role of this adapter is to be a switching hub and router installed in a file server. This board has several cost and performance advantages over a stand-alone switching hub/router. The adapter form factor is less expensive than a stand-alone box. Furthermore, the adapter accesses the server through the high speed (132 Megabytes/sec) and low latency PCI bus, as opposed to the stand-alone hub/router, which must access the server through a lower speed network link.

In this example, the on-board processor directs the traffic of the data packets in and out of the communication ports, the on-board memory and the PCI bus. This system could conceivably be implemented by installing four single chip communication adapters into a PCI file server, using the host PC's CPU to control the packet flow, but the performance would be far inferior to the intelligent adapter for several reasons.

First, the intelligent adapter's RISC CPU and its associate optimized code processes the packets faster than the host system CPU (e.g. 486, Pentium, Alpha or PowerPC). Second, processing the packets by transferring in and out of host system memory is extremely slow compared to transferring in an out of the intelligent adapter's on-board memory. Finally, a major benefit of an intelligent adapter is that it off-loads the host CPU from performing network processing functions, allowing the host CPU to concentrate on its file server duties, thereby increasing total system performance.

Other Intelligent Adapters Other applications benefit from an intelligent architecture for the same basic reasons; improved performance, improved functionality and reduced host CPU utilization. The diagram below illustrates an intelligent RAID controller. In this implementation, the SCSI controllers actually reside on a secondary PCI bus. The 80960 processor has its own private bus and memory connected to the secondary PCI bus through a PCI 9060 chip.

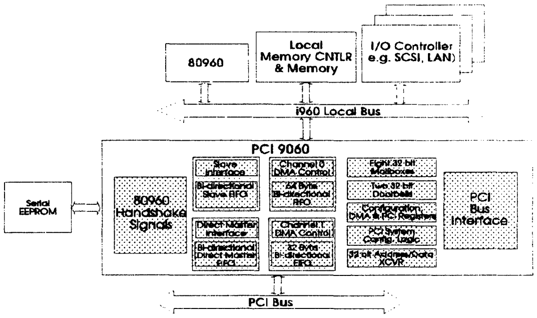


RAID Controller Adapter

PCI 9060 interface chip PLX Technology's PCI 9060 chip contains all the logic required to connect the PCI to a high performance adapter local bus. On the PCI bus side, the chip drives all the required PCI signals. On the local bus side, the chip provides a glue-less connection to Intel's 80960 processor family. Mode pins select one of three local bus configurations:

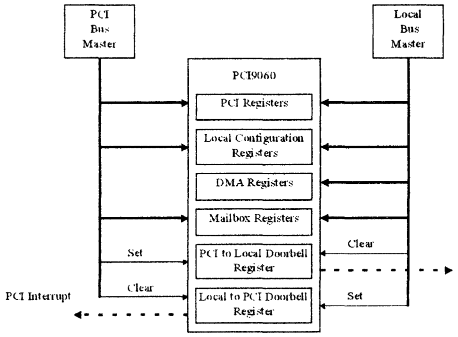
C/H mode	32 bit address/32 bit data non-multiplexed
J/K mode	32 bit address/32 bit data multiplexed
S mode	32 bit address/16 bit data multiplexed

The PCI 9060 can connect to virtually any local bus that matches one of the above configurations.

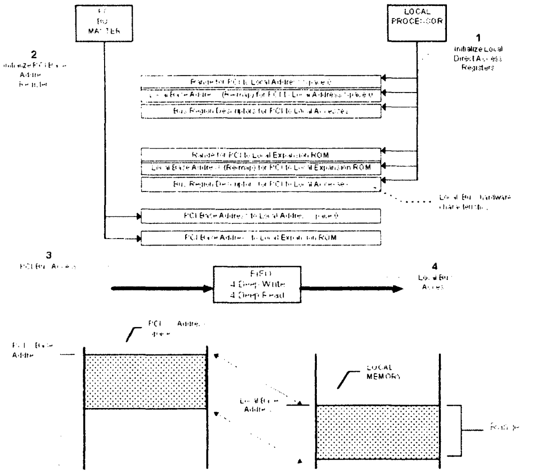


PCI 9060 bus interface chip

PCI 9060 Registers In addition to the register set required by the PCI specification, the PCI 9060 contains registers for programming the chip's slave, direct master and DMA data transfer modes. Eight 32-bit mailbox registers can be accessed from either the local or PCI bus for message passing. The two doorbell registers generate an interrupt to either the PCI or local bus when written to.



Data Transfer Modes In addition to message passing through the mailboxes, the PCI 9060 supports three data transfer modes, direct slave, direct master and DMA.


LOCAL PROCESSOR
1
Initialize Local Direct Access Registers
2
3
4
Range for PCI to Local Address Space 0
Bus Region Descriptors for PCI to Local Accesses
Range for PCI to Local Expansion ROM
Local Base Address (Remap) for PCI to Local Expansion ROM
FIFO
4 Deep Write
4 Deep Read
LOCAL MEMORY

Direct Slave Transfer

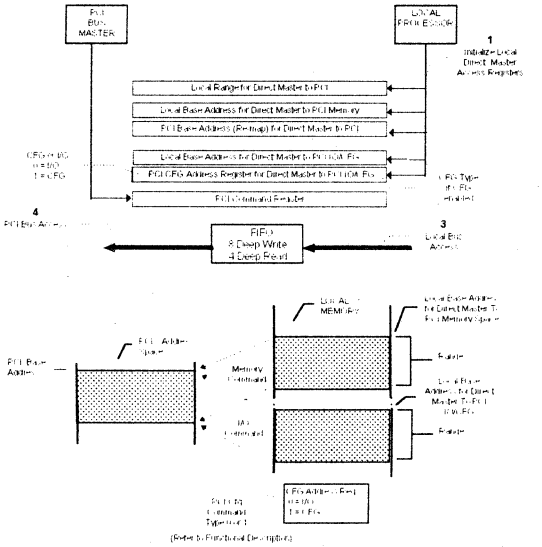
Direct Slave Transfer During configuration after reset or power up, the local configuration registers are programmed to provide correct address mapping between the PCI and the adapter local bus. These registers may be loaded from the local bus by the local processor, from the serial EEPROM or from the PCI bus.

When the master on the PCI bus provides an address intended for the adapter's local bus, the PCI 9060 arbitrates for the local bus and completes the data transfer.

The bi-directional FIFO in the chip allows efficient burst transfers between the local bus and the PCI bus, even when the two buses operate asynchronously.

In the intelligent adapter application, this mode is typically used to down-load instruction code from the host to the adapter's instruction memory. However, it may also be used to transfer data. The XPoint adapter employs their peer to peer software utility called "BusBIOS" that allows it to transfer data directly to and from other adapters in the system, without passing through the system host. Therefore, when the XPoint adapter is the target of another XPoint master adapter, it utilizes the direct slave mode for data transfer.

Direct Master Transfer In this mode, the processor or any other master on the local bus (e.g. intelligent I/O controller) may become a master on the PCI bus. In an intelligent adapter application this direct transfer mode is often used to allow I/O controllers to transfer data directly to and from host memory, instead of taking a less efficient route through the adapter's memory. Direct master mode is also used by the adapter's processor to fetch instructions directly from host memory or to configure other devices on the PCI bus.


PCI BUS MASTER
LOCAL PROCESSOR
1
Initialize Local Direct Master Access Registers
Local Range for Direct Master to PCI
Local Base Address for Direct Master to PCI Memory
PCI Base Address (Remap) for Direct Master to PCI
Local Base Address for Direct Master to PCI IO/CFG
PCI CFG Address Register for Direct Master to PCI IO/CFG
CFG or I/O
0 = I/O
1 = CFG
CFG Type if CFG enabled
PCI Command Register
4
PCI Bus Access
FIFO
8 Deep Write
4 Deep Read
3
Local Bus Access
LOCAL MEMORY
Local Base Address for Direct Master To PCI Memory Space
PCI Address Space
PCI Base Address
Memory Command
Range
Local Base Address for Direct Master To PCI I/O/CFG
I/O Command
Range
PCI I/O Command Type 0 or 1
CFG Address Reg
0 = I/O
1 = CFG
(Refer to Functional Description)

Direct Master Transfer

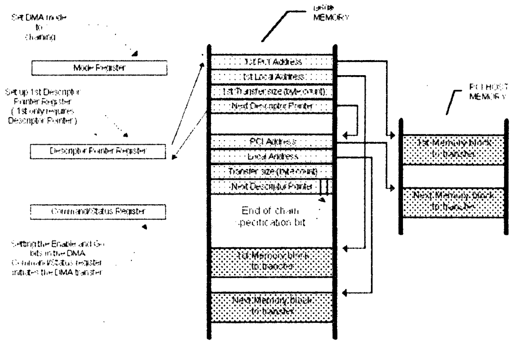
In a similar manner to the slave transfer, local configuration registers are programmed to provide correct address mapping between the PCI and the adapter local bus. These registers are loaded during initialization from the local bus, from the serial EEPROM or from the PCI bus.

In direct master mode the PCI 9060 can also execute I/O and configuration cycles, which allows the chip to configure other devices on the PCI bus.

When the local processor or I/O controller presents an address that maps to a PCI bus address (i.e. is not intended for the local bus), the PCI 9060 automatically arbitrates for the PCI bus and executes the data transfer.

The bi-directional FIFO in the chip allows zero wait-state burst transfers between the local bus and the PCI bus, even when the two buses operate asynchronously.

DMA Transfers The PCI 9060's two channel DMA controller provides the fastest and most efficient means of moving data between the adapter memory and host memory. In a typical intelligent adapter, this is the primary means for moving data. The PCI 9060 supports both chaining and non-chaining DMA transfers. A chaining DMA transfer is shown in the following diagram.


Set DMA mode to Chaining
LOCAL MEMORY
Mode Register
1st PCI Address
1st Local Address
1st Transfer size (byte count)
Next Descriptor Pointer
Set up 1st Descriptor Pointer Register (1st only requires Descriptor Pointer)
PCI HOST MEMORY
Descriptor Pointer Register
PCI Address
Local Address
Transfer size (byte count)
Next Descriptor Pointer
1st Memory block to transfer
Next Memory block to transfer
Command/Status Register
End of chain specification bit
Setting the Enable and Go bits in the DMA Command/Status register initiates the DMA transfer
1st Memory block to transfer
Next Memory block to transfer

Chaining DMA Transfer

To perform a chaining DMA transfer, the local processor programs the transfer parameters into the adapter local memory. These parameters include the PCI address and the corresponding local address, the transfer size in bytes and the descriptor pointers. The descriptor pointers tell the 9060 where to find the next group of transfer parameters in the chain. The local processor also loads the first descriptor pointer into the 9060 DMA registers and then initiates the transfer by setting the enable and go bits in the command/status register.

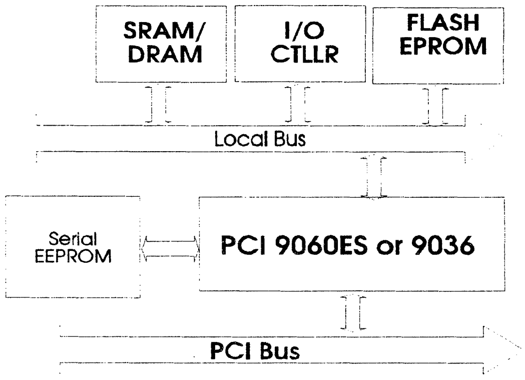
The 9060 then automatically completes the data transfer, including arbitrating for both the PCI and the local bus, moving the data through the FIFO, loading in the transfer parameters for each block and terminating the transfer. After the transfer is complete, the 9060 sends an interrupt to the local processor.

The advantage of chaining DMA is that it allows complex memory transfers to occur without requiring intervention by the local processor. This frees the local processor to complete other tasks.

PLX also supplies a low cost, reduced feature version of the PCI 9060 called the PCI 9060SD. This chip is similar to the PCI 9060 except it has no direct master interface and just a single channel DMA controller.

Direct Master Adapters

In the direct master adapter architecture, the I/O controller contains its own DMA controller and is therefore capable of transferring data directly to and from host system memory. Examples of such controllers include Intel's 82596CA Ethernet controller, National's MACSI FDDI controller, LSI Logic's ATMizer ATM controller.



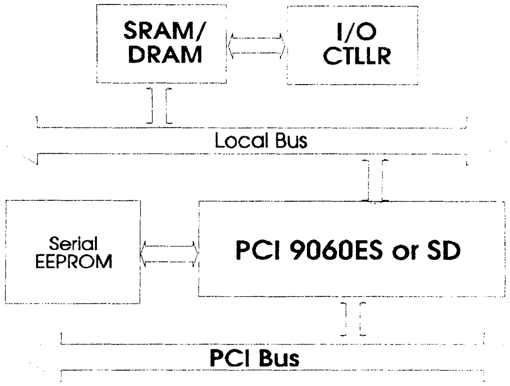
Typical Direct Master Adapter

PCI 9060ES and PCI 9036 interface chips
PLX offers two alternatives for implementing direct master adapters. The PCI 9060ES is identical to the PCI 9060, except it has no DMA controller. The chip transfers data primarily through the direct transfer mode described above. The PCI 9060ES is the best choice in cases where the local bus is asynchronous to the PCI bus. Because of its highly flexible slave interface (same as PCI 9060), the PCI 9060ES is also ideal for direct master adapters with memories that need to be accessed from the host.

The PCI 9036 is designed for direct master adapters in which the local bus is synchronous to PCI (i.e. can run at 33 MHz).

Slave Adapters

The typical slave adapter has a memory that connects to a simple I/O controller. The host accesses the memory as a PCI target. In a slave application either the low cost PCI 9060ES or PCI 9060SD may be used. They contain the same slave interface as the PCI 9060.



Typical slave adapter

HARNESS THE POWER OF PCI FOR EMBEDDED SYSTEMS

The PCI bus offers many advantages to hubs, routers, printer engines and other embedded systems that require a high-speed, low latency backplane. With data rates of up to 264 Megabytes per second, PCI gives embedded system designers the bandwidth previously provided only by proprietary architectures.

With PCI as a backplane, embedded systems gain all the benefits of using a high-volume PC-standard architecture: wide availability of low-cost I/O silicon (e.g. LAN, SCSI and graphics controllers), a proven standard architecture and compatibility with other manufacturer's hardware.

Embedded System Architecture

High performance embedded systems typically must transfer large amounts of data between I/O ports. In a switching hub, for example, the system performance depends on how fast the hub can move data between the LAN segments and how much traffic it can handle without saturating. High data transfer rates and low latencies are critical requirements that, until now, only proprietary buses could provide. Standard personal computer buses such as ISA(AT), Micro Channel and EISA could not meet these requirements.

PCI and Embedded Systems

PCI was designed specifically to improve bandwidth and latency in computer systems. However, the data transfer and latency benefits of PCI apply equally to embedded applications. Current versions of PCI support data transfer rates ranging from 132 to 264 Megabytes per second, comparable to or faster than proprietary buses in today's embedded systems. Furthermore, with PCI, latencies are fully under the control of the system designer and can be tuned to less than a few microseconds compared to tens or hundreds of microseconds for previous standard PC buses. Low latencies are critical for real-time processing, graphics and networking.

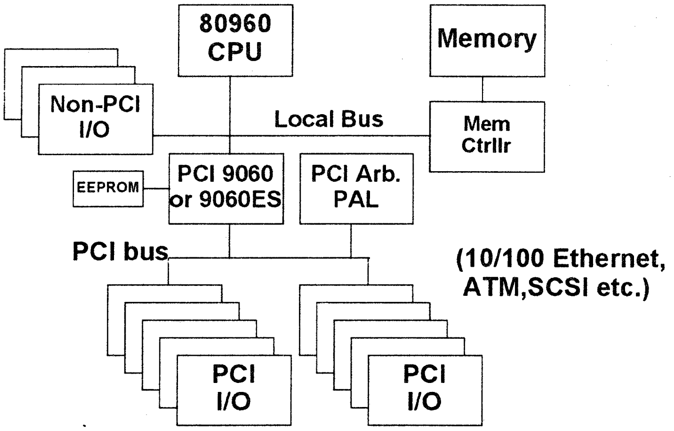
PCI Component Cost and Availability

Most suppliers of high volume I/O and graphics chips now offer a PCI interface in their components. Already, there is a wide availability of low-cost, high performance PCI I/O chips

PCI 9060ES Connects PCI to the Embedded System Controller

Until recently, the main challenge of implementing a PCI bus in an embedded system has been the lack of PCI controller silicon. There is a wide selection of silicon for Intel 486™ and Pentium™ processor-based PCs, but none for the unique requirements of RISC processor-based embedded systems.

The PCI 9060ES contains the logic required to generate and control a PCI bus in an embedded system. The PCI 9060ES is identical to the previously described PCI 9060, except it has no DMA controller. The following diagram illustrates a typical embedded system architecture:



Typical embedded system architecture

In this example, one of Intel's 80960 processors is the engine of the embedded system. The PCI 9060ES provides a glue-less connection to all members of the 80960 family including the H and C series (32 bit address/32 bit data non-multiplexed bus), the J and K series (32 bit address/32 bit data multiplexed bus) and the S series (32 bit address/16 bit data multiplexed bus). The PCI 9060ES may also be used with other similar processors.

The PCI 9060ES generates the PCI bus under the control of the processor. During initialization, the PCI 9060ES configures all the devices on the PCI bus from information in the boot FLASH or ROM which resides on the local bus. Configuration information includes the base address of the PCI devices, value of latency timers and other critical information.

The PCI 9060ES supports three data transfer modes between the PCI bus and local bus, where the processor and embedded system memory reside.

Direct Master Mode In direct master mode, the processor or other bus master device on the local bus, becomes a bus master on the PCI bus. Configuration registers in the PCI 9060ES map local bus address space to PCI address space. When the PCI 9060ES decodes that a local bus master is trying to access the PCI bus, it translates that local bus request into a PCI bus request and manages the data transfer between the local bus master and the selected PCI bus slave device. This is described in more detail earlier in the paper in the PCI 9060 description.

Typically in an embedded system, this direct master mode is used by the processor to configure the PCI and run-time registers of the PCI devices. FIFOs within the PCI 9060ES buffer the local and PCI bus, allowing them to run asynchronously.

Direct Slave Mode Most data transfer transactions occur when the PCI I/O device is the master and the embedded system memory is the slave. The reason for this is that most PCI I/O devices are designed to transfer data at high speed as masters. Typically they only act as slaves to accept configuration information. To accommodate this type of data transfer, the PCI 9060ES supports direct slave transfers, in which one of the PCI I/O devices is the master and the local bus device (usually embedded system memory) is the slave. Configuration registers in the PCI 9060ES map the PCI address space to the local address space. Deep FIFOs in the slave transfer path ensure efficient burst data transfers. (see PCI 9060 description earlier in the document)

If the local bus runs at 33 MHz and contains a zero-wait state system memory, the average data transfer rate is close to the theoretical maximum of the PCI bus. Of course if the local bus and memory subsystem are slower than the PCI bus, the embedded system performance will be determined by local bus performance.

Mailbox Registers As a third means of moving data, the PCI 9060ES also contains four 32 bit mailbox registers that can be accessed by both the PCI and local bus. If the PCI I/O device can support it, this is a means for message passing.

DMA Support As mentioned above, embedded systems typically do not require DMA support in the interface chip because most PCI devices are bus masters with built-in DMA controllers. However, some embedded systems have slave devices, such as memory, on the PCI bus. In these cases, the designer may elect to use the PCI 9060 chip instead of the PCI 9060ES. The PCI 9060 contains all the features of the PCI 9060ES. However, in addition, it contains a two channel chaining DMA controllers, programmed by the processor, which can transfer data at high speed between the slave PCI device and a slave on the local bus.

Arbitration and Interrupts The embedded system also requires a PCI bus arbiter. The design of the arbiter is application specific and not rigidly defined by the PCI spec. A simple arbiter may be implemented in a low cost PAL.

If there is more than one local bus master (i.e. a master besides the processor) on the local bus, the local bus will also require an arbiter circuit.

The PCI 9060ES contains a PCI to local bus doorbell register and a local bus to PCI doorbell register. When written to these registers assert an interrupt to the local bus or PCI bus respectively. Alternatively, the embedded system designer may choose to direct the interrupts from the PCI I/O devices through an OR gate directly to the processor.

The PCI 9000 series chips simplify the task of designing high-performance features into PCI adapters and embedded systems. The chips automatically handle complex data transfer functions in hardware, thereby freeing the designer from having to create these functions and minimizing host or local CPU intervention.

Product and Company names are trademarks/registered trademarks of their respective holders.

THE DESIGN OF A PCI FAST ETHERNET LAN ADAPTER

Liam Quinn
Thomas-Conrad Corp.
1908-R Kramer Lane
Austin, TX 78727
Ph. (512) 837-6155

ABSTRACT

With the addition of new users each day to the corporate networks, and the diverse range of applications run from the desktop PC environment, the capacity of existing LAN networks cannot handle the volumes of data, forcing the need for higher speed networks. As a result, there are a number of proposed high speed LAN technologies to meet this need. One such proposal is Fast Ethernet or 100Base-T, which is an extension of 10Base-T 10Mbps Ethernet. The development of the PCI Local Bus architecture with its 132MB per second transfer rate, makes it an ideal platform on which to design a Fast Ethernet LAN adapter. This paper describes the design of such an adapter and discusses the requirements of FIFO size and PCI bus latency. A brief discussion is also presented on the issues of cost savings, and ease of manufacturing and testability.

INTRODUCTION

There are several factors which heralded the development of high speed LAN networks. Primarily, the widespread use of a distributed client/server computing paradigm, the large increase of users on LAN networks, and the use of data intensive distributed application packages such as Lotus notes, graphics and document management application software and multimedia platforms. The development of new bus architectures, like the PCI Local Bus with its high bus bandwidth provides the platform for the development of higher speed networks.

There are several 100Mbps options which may be considered as a solution to those users seeking higher bandwidth on their network. FDDI, TCNS, Fast Ethernet and 100VG-AnyLAN. Fiber Distributed Data Interface was introduced about six years ago and is a well proven standard usually deployed as the backbone in the network, usually in the riser of a building interconnecting servers or routers, or between buildings separated by a highway, or street. It has not been used as a solution to the desktop, primarily because of cost, but with the emerging 100Mbps network solutions, its cost is expected to decrease. [1]TCNS is a 100Mbps proprietary protocol which runs on coax cable, Category 5 cable, fiber optic cable, and STP cable. It has been in production for over four years and is based on the Arcnet protocol, available in a full range of PC bus architectures. 100VG-AnyLAN is a proposed high speed network protocol which uses a demand priority media-access method, and features support for Ethernet and Token Ring frame formats. The 100Base-T high speed LAN proposal is an extension of 10Base-T, with 10 times the performance. This is achieved by a reduction in the bit time of each bit transmitted by a factor of 10. The packet format, packet length and management information remain unchanged. Similar to 10Base-T, 100Base-T is being standardized by the IEEE 802.3 committee and is based on the CSMA/CD media access protocol. There are three media specifications for the 100Base-T standard: 100Base-TX supports 2 pair Category 5 UTP cable, and Type 1 STP cabling. 100Base-FX supports multimode fiber optic cable, and 100Base-T4 which supports 4 pair Category 3,4 or 5 cable To ensure a smooth and seamless migration from 10Base-T to 100Base-T the Fast Ethernet Alliance (FEA) was formed by companies supporting the Fast Ethernet standard.

The Fast Ethernet Alliance

The Fast Ethernet Alliance is a multi-vendor group of companies who are committed to providing Fast Ethernet products and standard solutions. Their efforts are based on open standards, and products which are cost effective and interoperable. This ensures that the customers benefit from the interoperability testing of the hardware and software before they these products appear in the market. With the PCI compliance Checklist requirement for PCI based products, this PCI Fast Ethernet LAN adapter has been designed to meet the requirements of the Fast Ethernet Alliance interoperability specifications and the PCI Compliance Checklist.

PCI BUS LATENCY

The design and size of the FIFOs required on a PCI LAN adapter and a Fast Ethernet adapter in particular is a very complex and arbitrary task. The FIFOs are required to allow the adapter to maintain consistent data flow during PCI bus latencies when receiving or transmitting data through host memory. A primary objective is keeping the cost of the adapter to a minimum while at the same time optimizing the design and size of the FIFOs on the adapter. By definition the PCI Local Bus is a low latency, high throughput I/O bus. There are several mechanisms which predict and control worst case latency. There are three basic elements which form the relative latency of a PCI resource in order to become a bus master. These are the arbitration latency, acquisition

The **arbitration latency** is dependent on the arbitration algorithm of the host central arbiter, as well as the priority of the requesting PCI add-in module requesting bus master ownership.. It is also dependent on the present bus cycle. The arbitration latency is the amount of time it takes the central arbiter to assert **GNT#** in response to a **REQ#** from a PCI device requesting bus ownership. For the highest priority device this time will typically be 2 PCI clocks and the arbitration algorithm will be platform dependent. The primary elements controlling the present bus cycle, are the latency timer and disconnect termination.. These two PCI specific mechanisms are used to control the amount of time a bus master can stay on the bus when other requests are being asserted by lower priority PCI elements or expansion bus (ISA, EISA, MC) devices.

Each PCI bus master device has a programmable master latency timer which is cleared whenever that master is not asserting **FRAME#.** The master latency timer is reloaded each time the PCI bus master device asserts the **FRAME#** signal indicating the start of a cycle. The master latency timer count each rising edge of the PCI clock line during the time **FRAME#** is asserted, and expires within 256 or fewer PCI clock line periods.

The Bus acquisition latency is the amount of time the requesting PCI bus master must wait for the for the PCI bus to become free, after it receives the GNT# signal from the central arbiter. This is due to the completion of the present bus cycle by the current PCI bus master. The total time delay is between the requesting PCI device sampling the GNT# line on the next rising edge on the PCI clock line and the target PCI device sampling the FRAME# line asserted on the rinsing edge of PCI clock.

Target Latency is the amount of time between the target PCI device sampling FRAME# asserted and the assertion of TRDY# back to the PCI bus master. The total latency therefore is the is the total of the arbitration latency, the bus acquisition latency and the target latency. This value will vary from one system to the next due to the algorithms used on the system and the devices used in a particular system. This is a major concern for implementors of PCI add-in modules and in particular PCI LAN adapters. The key question is predicting the worst case latency in order to determine the amount of FIFOs on the adapter to minimize receive overruns and transmit underruns.

FIFO Requirements for the 100Base-TX Adapter

There are many factors which influenced the size of the receive and transmit FIFOs in the 100Base-TX LAN adapter. If the receive FIFO is too small it can result in receive packet overflow, and packet fragments are received. It the transmit FIFO is too small, it results in transmit underflow during the transmit of maximum sized packets (1500 bytes), and a fragment of the packet is transmitted. Therefore in the ideal case the FIFOs should be large enough to sustain transmit and receive data during the total access latency the adapter may encounter during PCI Local Bus accesses. The primary system factors which influence the sizing of the FIFOs are the PCI Local Bus I/O characteristics, the DMA arbitration state machine within the Fast Ethernet adapter, the host cache line width, memory subsystem. The size of the packets being transmitted an received are also important factor in deciding the size of the on board FIFOs. There is some measure of control over the transmit but not over the receive path, so the receive FIFO is typically larger than the transmit FIFO. Through simulation and analysis of large and small packets, it was found that for large packets the FIFO size required reduces as the packet size increases, but increased for smaller packets because of the increased overhead. The PCI Local Bus latency protocol described above provides for a measure of fairness and predictability, but there is no guarantee of a particular latency in the worst case. In a system where an ISA bus master accesses a PCI resource through a PCI/ISA bridge, the protocol of the ISA bus master allows it to retain ownership of the ISA bus and therefore the PCI bus indefinitely. There is no ISA bus master preemption protocol or way for the PCI/ISA bridge to terminate an ISA cycle. Similarly with an EISA bus the bridge cannot terminate an EISA cycle, but it does have a preemption protocol which restricts an EISA access cycle to 64 BCLK cycles: at 8.33mhz = 7680nsecs. The user or systems integrator should be aware of the different adapters in the system, when designing the system with a specific arbitration and acquisition latency time.

ADAPTER FEATURES

The PCI Fast Ethernet adapter features an integrated PCI dual protocol Ethernet MAC controller. It provides a direct connection to the PCI Local Bus interface and provides two ports for 10Mbps serial port and a media independent interface/symbol 100Mbps port. Each of the ports support full duplex mode. The PCI controller features two independent large FIFOs for receive and transmit packets, without additional on board memory required. The PCI controller has a powerful DMA controller with programmable burst size for low CPU utilization. The standard MII interface is provided for 100Base-T4 support, the scrambler and the PLS layer are integrated into the controller, providing for a more integrated and cheap 100Base-TX solution. The big and little endian byte order capability allow the adapter to be used in a range of host processor platforms.

Network Interface

The design uses a PCI controller which implements the 100Base-TX MII sublayer, the 100Base-TX physical coding sublayer, the 10Mbps and 100Mbps Ethernet MAC sublayers. The MII interface supports connections to 100Base-TX. 100base-FX and 100Base-T4 physical media specifications. The PCI controller supports two different modes to implement the physical connection to its MII/SYM interface. The first mode is the FEA compliant MII interface which may be used with the three media specifications defined above. The second mode is specific to the 100Base-TX mode. In this mode the PCI controller incorporates the PCS and the scrambling/descrambling function specified in the IEEE standard 802.3. This mode provides for a highly integrated, low cost 100Base-TX connection using the well defined twisted pair physical medium dependent layer specification (TP-PMD).

MII/SYM Interface

A clock generator and symbol device is connected to the symbol interface of the PCI controller. It implements the Physical layer portion of the TP-PMD standard, and is used to recover the receive clock, data recovery and NRZI conversion. The interface includes the five bits of transmit data and five bit of receive data as well as the 25Mhz recovered clock. It has a pseudo ECL interface port for an MLT-3 transceiver. There are three pairs of pseudo ECL lines between the clock generator chip and the MLT-3 transceiver: the TX, RX and SD. The transmit data lines are differential raised ECL signals and provide serial NRZI data at 125mhz to the MLT-3 transceiver. The receive serial data input pins are differential raised ECL signals and receive serial NRZI data from the MLT-3 transceiver at 125MHZ. The signal detect input signals are differential raised ECL signals and come from the MLT-3 transceiver indicating the receive serial data is valid. These three pairs of signals are very high speed sensitive lines and every effort was taken to keep these lines as short as possible, with each pair the same length. The transmit pair was routed on the opposite side of the adapter from the receive pair. Each pair was terminated at the receiving end of the lines, with the etch netted between corresponding pins of the two devices and then to the terminating devices, thereby ensuring the terminators were placed last of the etch run at the receivers. To avoid bit error sequences being transmitted on the network during power up and power down cycles, the design includes low power protection logic for the clock generator and symbol device and the MLT-3 transceiver. When the Vcc supply voltage crosses a predefined voltage level set by a resistive network, the transmit differential pair from the clock generator and symbol device are deasserted. It also forces the NRZI MLT-3 state machine to a quiet state.

100Mbps MLT-3 Transceiver

The MLT-3 transceiver device is the interface between the 100Mbps clock generator and symbol device and the 100Mbps port magnetics. It has a pseudo ECL interface, which operates at 125mhz, and connected to the clock generator and symbol device. On the output side the MLT-3 interface connects to the magnetics. The design uses filtered power and ground for the terminators associated with the 100Mbps differential transmit and receiver pairs. The design has minimum length of etch for all signals between the MLT-3 transceiver and the magnetics. The receive signal pair are routed on a separate layer from the transmit signal pair

10Base-T Network

The PCI controller provides a standard 7 wire 10Base-T port connection to an external endec and transceiver device. This is an integrated device and requires a minimum set of external components and magnetics. It requires an external 20mhz crystal to operate the transmit and receive clocks, and generates the transmit clock for the PCI controller. It supports full duplex mode, external and internal loopback modes. A set of general purpose pins is provided on the PCI controller that are used to control a variety of adapter functions.

Port LEDs

The design features a dual color LED to indicate link status and port activity. The link status LED is green when the link is good and red to indicate that there is link present. The port activity LED is amber and indicates activity on either the 10Mbps or 100Mbps ports. There are separate port LEDs to indicate which of the two ports is selected.

PCI Configuration Registers

The PCI Fast Ethernet adapter design supports the predefined header portion of the PCI configuration space, as well as sixteen internal command and status registers for host communication and adapter control. The configuration registers are used for identification, initialization and configuration of the PCI controller. The command and status registers are used for host communication, initialization , adapter feature set control and status reporting.

CSR Registers Functions:

- Serial EEPROM interface control register
- Frame descriptor control register
- WatchDog timer function register
- General purpose port register
- General purpose timer register
- Interrupt enable register
- Status register
- Transmit frame control register
- Receive frame control register

- Mode control register
- PHY port control register

The host communication interface manages the descriptor lists and data buffers which reside in host memory. The adapter PCI controller and the host resident driver communicate through two data structures: CSRs and descriptor lists and data buffers. The descriptors are the pointers to the receive and transmit frames in host memory and may be configured in a ring structure or a linked list of descriptors. There is a descriptor list for the receive frames and one for the transmit frames and each descriptor can point to a maximum of two buffers, which may not be contiguous in memory. Each buffer consists of either an entire frame or part of a frame, but does not exceed a single frame. The buffers contain only data with buffer status maintained in the descriptor.

Full Duplex Support

Traditional Ethernet operates in half duplex mode, where it can transmit or receive a packet, but not transmit and receive simultaneously. The MAC layer CSMA/CD protocol on Ethernet adapters regulates access to the LAN network by the attached node. By ignoring the carrier detect and collision detect scheme in the MAC, full duplex mode is possible, where simultaneous transmit and receive packets are on the network. In order to take advantage of full duplex mode, the adapter must be connected to a switch and in effect becomes the only node on the switched link. Of course the cost of the Ethernet switch is more expensive, making the choice of where to place the Fast Ethernet adapter in the network more critical. The PCI Fast Ethernet adapter design features a programmable full duplex mode bit that turns off the carrier and collision detect functions in the MAC chip. A critical use of this adapter with full duplex mode enabled may be in the server to switch network, where the servers are required to send and receive data simultaneously to clients. This feature allows the adapter to operate at 200Mbps which doubles the available bandwidth.

Serial EEPROM

The PCI Fast Ethernet controller provides a standard Microwire interface port for a 1024 bit serial EEPROM organized internally as 64 x 16 bits. It is used to hold the unique Ethernet address and any additional board specific information. This may include PCI vendor ID, initialization and diagnostic routines

MODULE LAYOUT AND PACKAGING

The design of the PCB layout for manufacturing, test and packaging is equally important when designing an adapter module. The cost of the complete LAN adapter must be insignificant compared to that of the PC host platform, and especially for PC vendors who may bundle the LAN adapter as part of the complete system. The cost of the raw PWB can be a significant factor to the overall cost of the completed adapter product. With the architecture of the PCI Local Bus interface, the restrictions of one load per interface signal, and the tight requirement placed on the etch length of the 32 bit address/data bus, the opportunity exists for compact module designs. Typically the PCB board vendors charge by the panel size, for example an 18" by 20" panel is common. By finding the optimum length and height of the PCB module and maximizing the number of modules per panel, there can be a significant saving on the cost of the raw PWB module. Given the maximum length of the short PCI raw module as 6.875", it is very possible to fit a LAN adapter design into a module much shorter than this. The height of the module is normally dictated by the number and type of media ports, and the ever present status LEDs. Another idea for smaller adapters is panelization for ease of manufacturing. These can be step and repeat in a row, or even two up modules in a reversed step and repeat format. This process is normally done at the PCB design stage prior to sending out the completed Gerber files to the PCB board shop. This results in groups of modules in a 'panel within a panel' of modules. The cost of the raw PCB is also related to the number of signal layers on the module. This design is a four layer module, but more complex multiport designs featuring possibly a PCI to PCI bridge will require a six layer module. The stencil is made to match placement data for the multimodule panel. During the manufacturing phase, the pick and place machine works on several modules simultaneously. This concept is carried right through assembly until the final stage where the brackets are placed on the adapters. By having the PCB vendor score the multimode panel for individual adapters, a simple snap in one or two directions separates the modules. The ICT test fixture may be designed to accommodate one or multiple adapters modules, and the same is true for systems or functional testing.

Power Distribution

With the high speed signals used on the PCI Fast Ethernet module, the design and layout of the module was an important consideration. To provide a noise free module it is necessary to have a noise free power distribution network, which includes Vcc as well as ground. For AC purposes Vcc is ground. The power distribution network must provide a return path for all signals generated or received on the module. The design objective is to deliver exactly +5V to each power pin on the module connector, regardless of the position of those pins, and the voltage should be free of all line noise. In the ideal case this could be represented as an ideal voltage source which has zero impedance. The zero impedance concept would ensure that the load and source voltage would be the same and would also mean that the noise signals would be absorbed since the noise generators have finite impedance. In actuality, a real power source has associated impedance in the form of inductance, resistance and capacitance, which are distributed throughout the power distribution network. Because of this impedance, noise signals are added to the voltage in the form of ripple or ringing. While these cannot be eliminated completely, the design goal of the layout and filtering is to reduce these as much as possible. The design uses a 10UF capacitor for each +5V entry point close to the edge connector. These larger capacitors filter out lower frequencies like the 60 Hz line frequency which are usually generated off the module in the system. A 0.01UF capacitor is placed across the power/ground pins of every active device on the module. In particular cases such as the PCI controller, a range of decoupling capacitors were distributed across the power and ground pins to reduce the effects of the various harmonic ranges. The adapter module design operates off the PCI +5V rail voltage, although the PCI controller operates on the 3.3V rail. This is generated using an on board voltage regulator and appropriate filtering. To avoid a single source of the regulator, given the demand by laptop computers, the design includes a universal regulator footprint to support either fixed or variable regulators from several vendors. A 10UF capacitor and a 0.01UF capacitor are placed across the +5V input and a 33UF capacitor is connected across the 3.3V output to stabilize and filter those voltages.

PCI Compliance Checklist

The PCI Fast Ethernet adapter has been designed to meet the PCI compliance checklist requirements as well as the FEA interoperability specifications. By using an internal island, the requirements to terminate each unused 3.3V pins on the module with 0.01UF capacitors, may be used with fewer decoupling capacitors. This technique works as long as the restrictions on the length of etch from the bus fingers to the terminators are met.

Summary

The most challenging aspect in designing a high speed LAN adapter, particularly for the PCI Local Bus is determining the optimum size of the receive and transmit FIFOs for the PCI controller to avoid receive overflows and transmit underruns. To deliver a low cost adapter, requires the PCI/MAC controller to integrate as much of the external logic in to the device. The development of software drivers which are highly optimized for performance are the crucial elements in differentiating one high speed LAN adapter from the next.

References:

PCI Local Bus Specification Revision 2.1

Solari, E.ISA & EISA Bus Theory and Operation. Annabooks, 1992

Yang et al., "FIFO Design for a High Speed Network Interface" in
Proceedings of the 19th Conference on Local Computer Networks

Fast Ethernet Alliance 1994. 100Base-T Fast Ethernet Technical Primer

IEEE 802.3 CSMA/CD Working Group Documents

82420/82430 PCIset ISA and EISA Bridges. Intel 1994.

Yang et al., "A Comparison of High Speed LAN Technologies" in
Proceedings of the 19th Conference on Local Computer Networks

Brown, C. and E. Holt, 1994. "FPGAs Lay Foundation for Customized PCI Interface,"
ASIC & EDA, August 1994, pp. 32 - 44

PCI-SCSI & PCI-IDE Solutions from Symbios Logic

Margit Stearns & Mark Winchell
Product Managers
Symbios Logic, Inc
1635 Aeroplaza Drive
Colorado Springs, CO 80916
Ph. (719) 573-3573 Fax (719) 573-3073

Symbios Logic (formerly NCR Microelectronics) discusses their family of industry standard PCI-SCSI products and their complete SCSI solution. A universal hardware and software architecture provides support for customers applications ranging from laptops to servers and host adapters. A second generation PCI-SCSI product will be introduced for the optimum in price and performance.

The first 16.7MB/s PCI-IDE Bus Mastering DMA was first demonstrated by Symbios Logic. This portion of the discussions will focus on a new, dual channel PCI-IDE solution offering the highest performance for multi-tasking OSes.

From PCI to SCI

David Gustavson and Qiang Li

SCIzzL
1946 Fallen Leaf Lane
Los Altos, California 94024-7206

Abstract

Is there anything more after PCI? Is PCI all one needs? This paper examines what properties are desirable for interconnects, determines that something additional is needed, and outlines a solution.

1: Introduction

PCI is the final ultimate universal standard bus, so fast and powerful that it meets every need.

PCI has been adopted by every significant vendor, after careful evaluation by their marketing departments.

If anyone ever wants more than PCI offers now, there will always be a faster PCI or a wider PCI or several PCIs.

There will be bridges from PCI to every previous bus, to provide backward compatibility with old equipment.

What more could one possibly want?

2: Desires

What does one want from an interconnect?

1) Low cost
2) Speed
3) Expansion
4) Multiple processors
5) Distance
6) Ease of use
7) Open standards

3: Frustrations

3.1: Low cost

Low cost requires narrow signal paths and a simple protocol so the interface logic and transceivers all fit in one corner of an ASIC. Low system cost requires interface behavior that is convenient for operating system and application software. For example, PCI systems typically need several bridges to divide the PCI bus into short sections, to keep the loading of each section within specification.

Because connecting through a bridge takes time and introduces deadlock hazards, PCI systems often "post" a write, where the written data are accepted by the bridge and the processor goes on without any confirmation that the written data actually reached their destination. This is often OK in a small reliable system, though it has obvious problems if an error should occur.

More subtle, however, are cases where the order of delivery really matters, where one has to ensure that certain data are really at the destination before some other data are read or written. Mechanisms for handling this complicate the bridge design and, worst of all, the software. Typically one designs the bridge to ensure that read transactions never pass write transactions, and then follows every critical write by a read to ensure that the data really have been delivered before proceeding.

This may be a tolerable expense in hardware and software if one understands the problem and designs for it from the start. But if these subtle issues are missed, the cost is probably a major chip redesign.

Other subtle problems are caused by the need for atomic operations and locks, essential for mutual exclusion and synchronization. If these are not well designed in the system architecture, they can be very difficult to handle in systems containing bridged bus segments.

3.2: Speed

Bus speeds can only be increased by making them shorter or wider. *Shorter* (in length and/or fewer sockets) makes buses less useful, and PCI has already reduced that dimension about to the practical limit.

Wider makes buses more expensive, because more transceivers and pins and space and power are needed. Worse yet, doubling a bus width does not double its speed, because of the inherent overheads of arbitration, connection, addressing, and disconnection, and because of the limitations due to skew.

Skew refers to some data bits arriving before others across the width of the bus. Bus cycles must be long enough that the receiver can keep each cycle's bits segregated correctly, unless the protocol supports automatic skew compensation, which is expensive and adds its own overhead time. The skew problem becomes more severe as bus width increases.

So, hoping for much faster versions of PCI is unrealistic.

3.3: Expansion

Expansion capability requires length and more sockets, which don't come easily for PCI—expansion requires bridges, which are, of course, additional components and introduce a variety of problems, some of the hardest of which were mentioned above.

PCI handles modest expansion needs, but greater expansion requires an interface to something else.

3.4: Multiple processors

Microprocessors have become so inexpensive that we often wish to use many of them in one system. Unfortunately, they tend to be independent prima donnas that do what they want when they want, and making them cooperate efficiently in a system requires imposing discipline.

An interconnect for multiprocessor use must support mutual exclusion, to force the processors to perform certain operation sequences one-at-a-time.

It must also have well-defined atomic behavior, so that software can assume certain variables are written all-at-once, not a few bytes at a time. (It's hard to imagine a worse error than reading an address variable, or pointer, that is in the process of being updated by another processor, and getting some of the bits from the new in-progress write with the others left over from the previous write!)

It must provide end-to-end feedback when needed, so that the correct sequencing of writes and reads can be ensured when necessary.

3.5: Distance

Distance is becoming a greater and greater problem. This is surprising, since everything in this industry shrinks every year as integration technology improves. However, systems tend to remain a convenient size for human beings—the improved technology is used to get more capability into the same space, not to reduce the space occupied by keyboards, displays, desks, and offices.

As technology advances, clock speeds increase but the speed of light stays the same. This means that it takes more clock cycles each year to access information that comes from some distance, whether from the next chip, board, box, office, or building.

All the tricks we know to compensate for distance involve caching. I.e., whenever we request information from a distance, we request more than we need at the moment (a cache line instead of a byte), and we keep a copy of this information locally in a cache memory. It is more efficient to transfer longer blocks, because the overhead costs are spread over more data, and if the same data are used again soon (temporal locality) or if nearby data are needed (spatial locality), then filling a whole cache line is a good investment.

Smart compilers or smart programmers can often anticipate which data will be needed, and can cause cache lines to be prefetched so that the data are in the local cache already when they are needed.

But caching data creates a very serious problem: each cached copy is a duplicate; whenever the data are modified all the other cached copies become incorrect. The duplicates must either be corrected or discarded, or serious logical inconsistencies will result. This is the cache consistency or *cache coherence* problem.

It is possible to solve this problem by using software to flush the affected cache lines when necessary, but that is complex and usually very slow.

What one really needs is for the interconnect hardware to monitor the use of shared data and keep the caches consistent. That isn't hard, if it is designed into the system architecture, but it is very hard to add as an afterthought.

3.6: Ease of use

Ease of use requires an architecture that provides a simple clean way to do whatever needs doing, so one doesn't have to improvise and kludge and deal with unintended consequences for every new project.

The architecture should provide building blocks that are versatile and well-behaved, and it should be scalable so that it can be applied to a very wide range of problems.

The architecture should be free of arbitrary limitations like length and loading, and should provide a wide range of cost/performance tradeoffs that remain compatible.

Having a single modular architecture that covers many needs results in accumulated expertise and lower design costs. These savings can be applied to doing a better job of interfacing the product to its end user.

3.7: Open standards

Standard interfaces are the key to competition, leading to low cost, a large and stimulating marketplace, innovation, and versatility.

Open standards are those not dominated by any single company, so the user can have confidence that there will be real competition among multiple vendors, leading to lower prices and a greater variety of products. Vendors need to know that no other vendor has any special advantage that may be impossible to compete against.

Standards organizations like the IEEE and ANSI in the US and the ISO/IEC internationally have evolved over many years to meet the need for fair and impartial standardization, and continue to evolve to keep up with the increasing pace of technological evolution.

Sometimes company-defined standards are declared "open" to gain wider acceptance and create a larger market, and sometimes these do successfully make the transition to become truly open standards.

4: Pleasures

A good solution to these problems has been developed, has completed the standardization process in the US (ANSI, IEEE Std 1596–1992[1]) and will soon be published as an international standard (ISO/IEC). Initially called "SuperBus," reflecting its original goal of being a super-fast far-future hyper-Futurebus, it was soon obvious that it is fundamentally impossible for any bus to provide enough speed over practical distances to support significant multiprocessing with today's fast processors.

The solution was to not use a bus at all, but to provide bus-like services by communicating over a potentially large set of independent point-to-point links. The name then became the "Scalable Coherent Interface," or SCI, which unfortunately brings no useful image to mind for most people. Recently people have begun to call this system the Local Area MultiProcessor, or LAMP, which brings to mind LANs, multiprocessors, communication, data sharing, and campus-size distances, a more useful mental starting point.

Happily the "far-future" arrived in 1991. Once the traditional "bus" mindset was overcome, progress was extremely rapid. There aren't many solutions to these problems that have the right scaling behavior, so most decisions were easy.

Since 1991, work has continued on verification, polishing, prototyping, testing, and extending the SCI standard in several directions.

One of these extensions is software-oriented, to greatly simplify sharing data in a heterogeneous multiprocessor. ANSI/IEEE Std 1596.5[2] allows the user to describe the shared data so precisely that the compiler can automatically convert it for correct calculations, even in systems that mix processor word sizes and endianness, as long as the interconnect obeys the principle of "address invariance." If the hardware violates address invariance by swapping bytes to reformat the data on the fly, the way many interfaces used to do, this problem becomes extremely complex because the hardware does not have enough information to always do this correctly. Software then has to undo damage caused by the hardware's clumsy attempts, which requires knowing the details of the path taken by the data and the settings of the hardware options along that path at the time the data were transferred.

Other extensions (nearly complete) include a new electrical signaling standard for low-voltage differential signals (LVDS), P1596.3, and a 500 Mbyte/s memory-I/O-processor interface (RamLink), P1596.4.

Ongoing work includes generating a guide for switch and bridge design, P1596.1; optimizations for kiloprocessor systems, P1596.2; optimizations for improved Real-Time schedulability (SCI/RT), P1596.6; and three new projects just starting, for standardizing improved cables and connectors, fiber-optic parallel ribbons, and a hyper-fast memory chip interface (about 2–4 Gbyte/s).

In 1994 IBM demonstrated an SCI link interface chip, biCMOS, running 16-bit-wide links at 1 Gbyte/s, and implementing automatic signal deskewing. A 125 Mbyte/s CMOS interface chip that includes transceivers, protocol logic, and fifos is available from LSI Logic. Convex/HP is shipping the SPP-1000 Exemplar, which uses GaAs SCI interfaces internally to support 128 PA/RISC processors.

Many other computer designs based on SCI are under way though still unannounced, and several new chips are expected shortly.

The Navy, Marines and Air Force selected SCI as the interconnect for their Joint Advanced Strike Technology, instead of the 6 different interconnects used in the F-22.

And best of all, of course, bridges are being built to connect PCI and SCI.

4.1: PCI–SCI bridges

These bridges make it possible for PCI systems to use SCI to escape some of PCI's limits, and they allow SCI systems to access PCI devices.

A bridge to SCI can't solve every problem, of course—if PCI conceals what is happening in a processor's cache, SCI can't keep that cache consistent with others and thus software techniques or other workarounds (like never caching data from certain address ranges in the processor) will have to be used.

And obviously a bridge can't transfer data faster than the PCI bus it is connected to, or solve ordering, atomic, or mutual exclusion problems that already exist between the processor and the bridge.

Still, the bridge to SCI will greatly increase the reach and the power of many PCI systems.

5: Mechanics

So, how does SCI perform these feats?[5,6]

5.1: Links

SCI replaces the shared bus with unidirectional point-to-point links to eliminate the fundamental physics problems of busing. Narrow links are compatible with integrating the entire transceiver set and protocol logic and user interface on single chips, which in turn reduces skew problems and greatly lowers costs compared to multichip solutions. Differential electrical signals are used for low noise sensitivity, low noise generation, and high speed. Since the links are unidirectional, these can easily be converted to optical signals for use in fiber ribbons such as Motorola's Optobus. The point-to-point links are independent, allowing for concurrent transmission by many processors or I/O devices at the same time.

The protocols are (mostly) self-timing, and are independent of cable or fiber length. Of course, increased distance reduces system performance, but only for those transactions that use the long links, and far less than would be the case for bus-style protocols.

For highest performance, these links are connected by a switch that routes information from one device (called a node in SCI) to another. However, switches are expensive so extra features in the link protocol allow nodes to be connected in rings by simply daisy-chaining output links to input links. That offers very low cost, but reduces performance because links in a ring carry information for other nodes as well. Rings are also simple and inexpensive for a motherboard or backplane, needing few layers.

A wide range of cost/performance tradeoffs is possible by mixing switches and rings, or by using meshes of rings.[4] Low-end systems can start with rings and add switches as demands increase. Systems can grow easily over time, because there are few constraints on topology or distance.

5.2: Protocols

To provide bus-like services, SCI protocols send request packets and response packets over these links, rather like a split-response bus. These packets are smaller and simpler than the packets used in LANs, and the protocols are far more efficient than LAN protocols. This is possible because the entire connected system in SCI shares a single flat 64-bit address space, and because the links run continuously so that no time is lost for synchronizing the receivers at the start of each packet.

The high-order 16 bits of the 64-bit address are used for routing packets, and are placed in the front of each packet. Thus the routing information is available in just a few ns for efficient use by switches and bridges. Though a switch or bridge keeps a copy of each packet temporarily in case it must be retransmitted, the packet can be forwarded on out the other side without waiting for it to arrive entirely. This "cut-through" routing greatly reduces latency.

When a packet reaches its destination or a buffer in an intermediate switch or bridge, it is removed from the ring and copied into a buffer. If there is not enough space in the buffer, the packet is discarded. In either case, the packet is replaced by its "echo" packet, which proceeds on out the outgoing link and returns to the previous sender, who still keeps a copy of the packet. The echo either confirms delivery to the next buffer, so that the saved packet can be discarded to make room for new traffic, or the echo calls for the packet to be sent again until it is delivered successfully. The echo provides local hop-by-hop flow control from bridge to bridge, but not end-to-end confirmation.

Confirmation of the final delivery of a request to its destination, and completion of the requested action, is made by a response packet. The response packet is routed through a large system hop by hop in the same way as the request, though it may follow a different path.

If the request was a read, the requested data are carried by the response. If the request was a write, the data were carried in the request and the response merely confirms delivery. A processor may choose to continue after a write without waiting for confirmation (weak ordering), or it may wait (strong ordering). Thus ordering models are implemented by the processor's interface to SCI, and by its instruction set. SCI does not restrict these choices.

5.3: Cache coherence

SCI includes protocols for keeping caches consistent[7]. Since there is potentially a great deal of concurrency in an SCI system, it is impossible to use the snooping techniques that have become common in bus-based systems. Snooping effectively requires all transactions to be visible to each cache controller, which severely limits performance and also places a high premium on making the cache controllers very fast (expensive). Bridges that join two snooping systems must fully account for the caching on both sides, making them expensive.

SCI's coherence protocol keeps track of every cached duplicate of each cache line in memory, by building a linked list of sharing caches. When the data are modified, this list is used to locate all the now-incorrect copies and delete them from their caches (invalidation). In many cases the shared data are no longer being used, so discarding them from a cache improves cache efficiency by making room for more-useful data. If a processor is still using the data, its next access results in a cache miss and a fresh copy is then requested from memory.

When data are requested from memory, the request packet includes the (16-bit) return address for use in routing the response. The request also specifies whether the data might be modified or will be kept read-only. The memory controller uses this information to keep track whether its own copy of the data is current or not. If its data are current, the response carries them to the requester. In all cases, the return address from the request is swapped with the return address from the previous request, which has been saved by the memory controller. This requires 16 bits plus a few state bits for each 64-byte cache line in memory.

If the response does not include the data, the requester uses the pointer to the previous requester, knowing that it must get the data there instead of from memory. Thus contention at the memory is avoided—any request visits memory only once.

The previous requester responds once it has the data and is ready to share (perhaps has finished modifying it). That response also completes a successor link between the two nodes, resulting in a doubly linked list of nodes that are sharing cached data. Thus SCI cache controllers contain two 16-bit pointers for each cache line in addition to the usual tag (the line's memory address) and status bits. This way of distributing the cache sharing directory scales correctly for all sharing patterns, whether a line is unshared or is shared by 64K nodes. It also distributes the traffic related to maintaining the directory, so that it does not contribute to contention at the memory.

One node is in a unique position in this list, being doubly linked to its predecessors (if any) but only singly linked to its successor (if any). That node is called the "head," and is the only node in the system that may modify the cache line (and then only if it requested its copy from memory with permission to modify). Thus there is a well-defined order in which nodes get access to a cache line, and exactly one node that can modify the data at any time. When the head modifies the data, it also follows the doubly linked list to its predecessors and notifies each of them that their copy is invalid, and removes them from the sharing list. The head's processor may wait before continuing to execute instructions after this write instruction until all the invalidations are complete, or not, depending on the ordering-consistency model that is being enforced by the processor for that write.

If a node runs out of cache space, as often happens, it can free a line by invalidating it (roll-out) and then leaves the sharing list by telling its neighbors to point to each other. If a node needs to modify data that it was previously sharing, it rolls out and requests the data from memory again. Sometime later it will become the exclusive head and can only then modify the data. Note that the data may have been modified by others meanwhile, so it may be quite different from the shared copy it had earlier.

5.4: Forward progress and deadlocks

The sequence enforced by this process is important for guaranteeing forward progress. Forward progress guarantees are critical in multiprocessor systems. Without them, it is likely that some processors will become starved and make no progress until others are finished, serializing what should have been a parallel process.

Deadlocks are even worse, bringing progress to a halt and wasting resources until someone intervenes.

SCI has taken great care to avoid deadlock hazards in its protocols and to ensure forward progress. For example, buffer space for receiving packets in an SCI node is subject to a simple reservation mechanism that grants immediate access if there is no congestion, but reserves space when necessary to prevent starvation.

An important property of SCI's coherence protocols is that all communication is performed using ordinary addressed packets. Thus, a bridge or switch does not have to know anything about cache coherence, all it has to do is look at the packets it sees and route them as usual.

Furthermore, there is no special premium on making cache controllers fast. As long as they do the right thing with the packets they receive, and send the right packets when they need to, the caches will be kept consistent. Of course, no one likes slow devices—but in SCI a slow device only affects the parties that use it, while on a bus with snooping coherence protocols the system speed is set by the slowest cache controller. This is a significant economic effect.

5.5: Locks and mutual exclusion

It is usual in bus-based systems to seize the bus briefly to exclude all other communication, and then perform a read-modify-write to implement locks and mutual exclusion. That strategy only works when all information flows through one bus, often causes trouble when the system includes two-port memories or bridges to other buses (even processor-chip buses—designers often forget that the processor interface is actually a bridge).

A high performance scalable system cannot shut down the whole communication system to do locks and synchronization, as the resulting performance would be extremely poor. Instead, the action of the mechanism must be limited to one point, either the processor or the memory controller (of course, either of these parties might in a particular case actually be an I/O device).

The cache coherence mechanism can be used to get an exclusive modifiable cache line, which can be held briefly and updated by read-modify-write techniques before the interface allows the successor node to have a copy.

If the lock variable is not being cached, then the processor sends an explicit lock command with appropriate data in its request. When the memory or I/O controller acts on that request, it does whatever is necessary to assure exclusive access (e.g. briefly locking out other ports) while it carries out the read-modify-write internally. Then it returns the result in a normal response packet.

Lock packets look like ordinary requests and responses: switches and bridges do nothing special for locks.

The lock commands that are defined in SCI include swap, compare&swap, and fetch&add. The first two are critically important for allowing multiple processors to add items to a linked list while one server is removing items. Doing this without using software semaphores is important for performance and for reliability. Fetch&add is often used for distributing work among multiple processors and for implementing barrier synchronization to keep them working in step.

5.6: Atomic transactions

SCI (by referencing a related standard, ANSI/IEEE 1212, Control and Status Register Architecture[3]) defines a set of transactions that are to be executed atomically.

Thus software can rely on reading certain data reliably, without any hazard that another processor might be writing it in sections at the same time.

6: Conclusions

Using multiple microprocessors in a system is very attractive and important, but requires some forethought to be inexpensive and efficient. Hardware and software issues must both be considered by the system architect.

PCI will be a widely available interface, but it was not designed to work well in a multiple-processor system.

SCI has laid the groundwork for the Local Area Multi-Processor model of computing, which will soon become available from multiple vendors. Though LAMP superficially resembles clusters or Networks of Workstations (NOW), LAMP supports true distributed shared memory, the most attractive model for multiprocessor software, with cache coherence enforced by hardware.

Message-passing is very fast in such a machine, so programs written for or already converted to run on today's multicomputers will run very efficiently on a LAMP.

But converting programs from their single-processor prototype versions to run efficiently on multiple processors is far simpler in a LAMP than in a multicomputer that only supports message passing, without shared memory.

First, the original program can run on a single processor, which may be the workstation where it was developed or may be a very fast workstation-like processor inside a high performance supercomputer cabinet.

Next, the program can be recompiled with automatic parallelization enabled. With today's compilers, that may not do very well, but the technology is improving rapidly now that real LAMP machines are available (Convex).

Then hints and compiler directives can be used to improve parallel performance, and gradually parts of the program can be rewritten and optimized.

The great thing about this is that the program can continue to do useful work throughout the whole conversion process, unlike the conversion to message-passing, which requires a major restructuring of the program to get any advantage at all from parallelization.

Networks have new opportunities in a LAMP. First, they can continue to use their traditional protocols but use the shared memory to move data at very high speeds. This approach will remain limited by the overheads of the protocol software layers, but the backward compatibility will be important and valuable to many users.

Next, they can move to simpler protocols that take full advantage of shared memory. Establishing a connection involves negotiating memory access rights, after which the shared memory reduces protocol overhead to its absolute minimum.

LAMP also supports scalable distributed shared I/O. E.g., disk drives attached to one node could be accessed directly from any other, with little software intervention. Neighboring nodes can pool their drives in disk arrays for high performance fault-tolerant parallel I/O. This is particularly attractive for high performance database servers.

SCI was made very simple so that it could run fast, using RISC-like principles. Alternatively, that simplicity can be used to get low cost, by sacrificing some speed—one doesn't need SCI's Gbyte/s links to keep up with today's PCI interfaces.

7: For further reading

1. IEEE Std 1596-1992, Scalable Coherent Interface. Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, 800-678-4333.
2. IEEE Std 1596.5-1993 Shared Data Formats Optimized for Scalable Coherent Interface (SCI) Processors. *ibid.*
3. IEEE Std 1212-1991, Control and Status Register (CSR) Architecture for Microcomputer Buses. *ibid.*
4. H. Bryhni and Bin Wu, "Initial Studies of SCI LAN Topologies for Local Area Clustering," First International Workshop on SCI-Based High-Performance Low-Cost Computing, Santa Clara, California, August 17–18, 1994, pp. 71–76.
5. D. B. Gustavson, "The Scalable Coherent Interface," *IEEE Micro,* Feb. 1992, pp. 10–22. An introduction to SCI, superceded by [6], but listed here because it is readily accessible.
6. D.Gustavson and Q.Li, "Local Area MultiProcessor: the Scalable Coherent Interface." in *Defining the Global Information Infrastructure: Infrastructure, Systems, and Services,* S.F. Lundstrom, ed., SPIE Press Vol. 56, pp 131–160 (1994).
7. D.V. James, "SCI Cache Coherence," *Cache and Interconnect Architectures in Multiprocessors,* ed. M. Dubois and S.S. Thakkar, Kluwer Acad. Pub., Boston, 1990, pp.189–208.

For more information about SCI or the current status of the SCI–PCI bridge projects, contact:

David B. Gustavson, Executive Director, ***SCIzzL***
1946 Fallen Leaf Lane, Los Altos, CA 94024
415-961-0305, fax 415-961-3530
email dbg@sunrise.scu.edu

SCI ENABLED PCI EXPANSION

Antonio Turgeon
Dolphin Interconnect Solutions, Inc.
3625 East Thousand Oaks Blvd., Suite 216
Westlake Village, CA 91362-3626 USA
Phone: (805) 371-9493
E-Mail: Dolphin_USA@delphi.com

Abstract

The latest generation of high-performance RISC processors on a chip are providing phenomenal CPU performance characteristics. Concurrent with these enhancements, users desire to be able to economically expand systems, incorporating high-performance CPU's, very large memory address space, and increased I/O bandwidth, while still maintaining existing investments in software and training. Translation of these performance enhancements into system throughput has typically been constrained by the limitations of conventional computer bus technologies.

Peripheral Component Interface (PCI) is a relatively new bus technology standard providing a synchronous, processor independent, multiplexed 32-bit architecture, with extension capability to support a 64-bit wide bus. The PCI bus provides for bus speeds as high as 33 MHz with transfer rates as high as 132 MBytes/sec via burst mode. This standard provides full multimaster capability, hidden central arbitration, concurrency with processor memory subsystems, write-back and write-through cache support, and automatic configuration of PCI add-in cards at power-up. Most of the major computer manufacturers are planning to provide at least one PCI expansion slot in future product releases. Any PCI-compliant peripheral should work with any PCI based system.

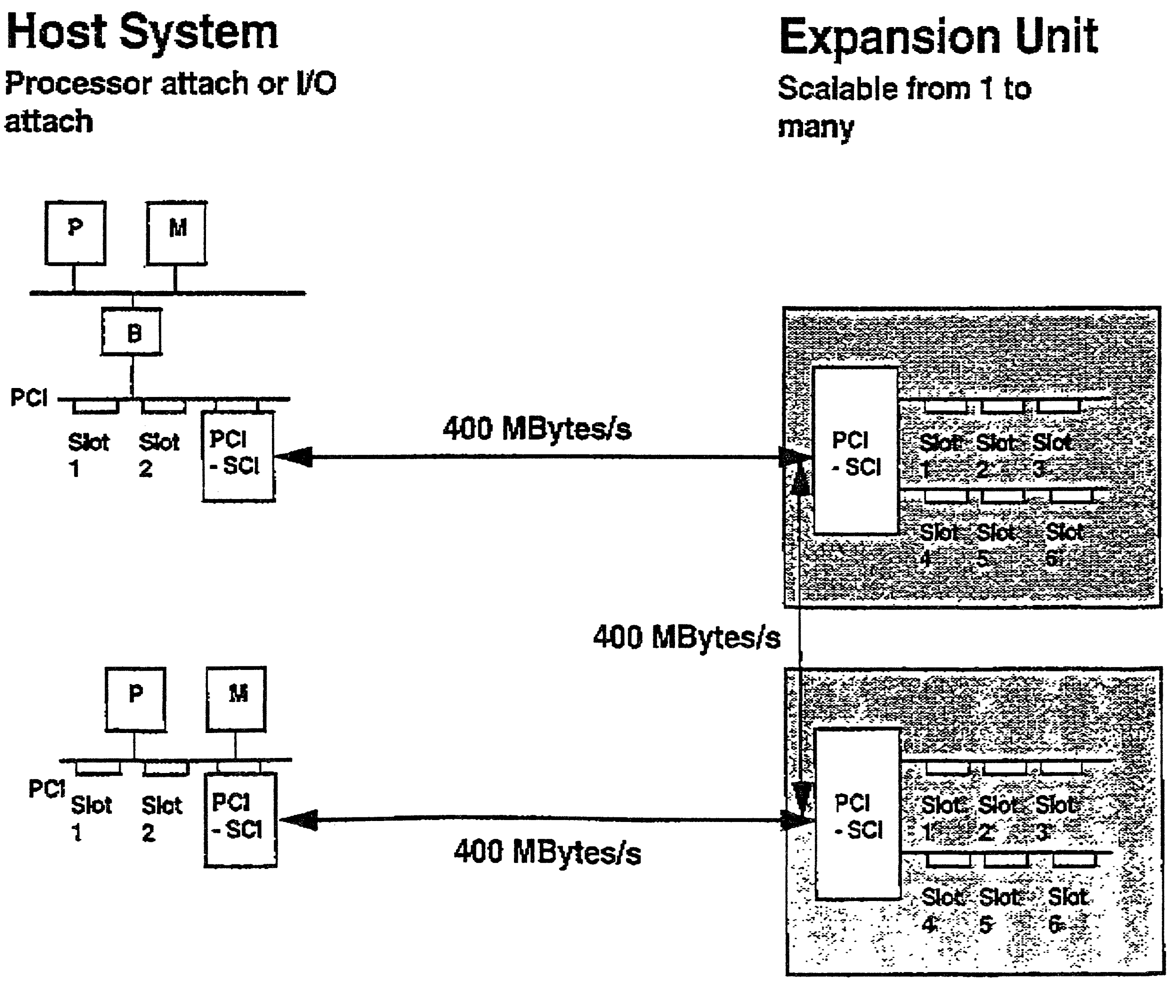
Scalable Coherent Interface (SCI), ANSI/IEEE standard 1596-1992 is a new technology established as an industry standard and defined specifically to break the speed and distance limitations associated with conventional computer bus technology. SCI defines an architecture of distributed shared memory with optional caching that interfaces microprocessors, workstation clusters and large I/O systems. The specification utilizes unidirectional point-to-point links to overcome traditional transmission line practicalities and offers extremely high interconnect performance (up to 1 GByte/sec data transfer rates), independent of the use of the shared memory features. By definition, SCI technology enables scalable architectures with ample room for next generation systems by specifying up to a 64-bit address and up to 64,000 nodes.

The discussion overviews general design considerations and SCI characteristics and features that provide solutions to existing limitations of conventional bus technologies. The area of application interest is concerned with PCI expansion capabilities implemented via a high-performance PCI-SCI bridge interface between the SCI and the Peripheral Component Interface.

FEATURES

Conforms to PCI Local Bus Specification rev. 2.0.
Supports non-coherent SCI transactions.
Supports PCI-PCI bridges over SCI.
Exports configuration cycles.
Built-in DMA unit.
Write gather, assemble 64 bytes packages to SCI.
Read prefetch, read whole cache lines from SCI.
Immediate write transfer acknowledge from PCI/SCI.
Generates SCI lock types.
Accepts locks from SCI and generates PCI lock cycle.
Address translation, 32 bit PCI to full 64 bits SCI.
Generates non-translated 32 bits address on PCI.
On-chip Address Translation Cache (ATC).
Retry on PCI bus on reads to free bus.
Accepts all write_sb packets, read_sb, nread64.
Generates write_sb and read_sb of any size (1-16).
Supports up to 8 interrupt input lines with programmable priority.
Built-in arbiter for 8 masters.
PCI Configuration space available from SCI.
CSR in conformance to IEEE Std. 1212-1992.
CSR available from both SCI and PCI.
JTAG (IEEE Std. 1149.1) implementation.

PCI expansion system



PCI Expansion Solution Objectives

- **Improve the typical limited scalability from a maximum of 4 slots per PCI bus.**
- **Offer scalability solution providing for many PCI buses.**
- **Reduce the typical on-board bridging constraints of PCI-PCI bridge chips.**
- **Provide expansion flexibility with internal or external cabinets.**
- **Provide High Interconnect Bandwidth.**
  SCI Links from 200 MBytes/sec B-Link 400 MBytes/sec.
- **Provide High Bridge Bandwidth - Throughput up to 80 MBytes/sec.**
- **Provide Low Latencys: Node to Node 0.15µs; Round trip read < 5µs**
- **Provide High Scalability - Many host expansion unit connections.**
- **Provide transparent read/write PCI register accesses.**
- **Provide a transparent bridge, summarized as follows:**

	PCI	**SCI**
♦ **Processor independent**	√	√
♦ **High Bandwidth, low latency**	√	√
♦ **Shared Memory**	√	√
♦ **Locking Support**	√	√
♦ **Scalability**	**Low**	**High**
♦ **Distance**	**Inches**	**Meters**

PCI-SCI Bridge

Typical PCI-PCI bridges are devices that make it possible to extend a local PCI bus with an additional PCI bus on the same circuit board. Using PCI-PCI bridges a multilevel bus hierarchy can be built allowing for more loads on the same bus domain.

The PCI-SCI Bridge allows having PCI devices connected to a host or multiple hosts over SCI. Many PCI I/O expansion nodes can be maintained and accessed by a host attachment connected to SCI, creating the equivalent of peer host bridges. The PCI-SCI bridge implementation utilizes two ASIC's, the SCI Transport Layer Link Controller ("LC") and the PCI Protocol Layer ("PtoB"). The Link Controller is Dolphin's latest CMOS version of the SCI link layer. Both the LC and PtoB incorporate the 400 MByte/sec B-Link. The B-Link connects components with an SCI Local Bus. The PtoB connects to the PCI bus on one side and B-Link device(s) on the other.

The PCI-SCI bridge will utilize PCI burst capability to build SCI packets that effectively utilizes SCI bandwidth. The protocols include hardware support for busy retry, guaranteed data delivery, error checking, flow control and a subset of lock transactions is also supported to enable hardware locking in a shared memory environment. An event reporting mechanism is used to report interrupts, errors and other events from the PCI -SCI bridge to the host attachment.

The PCI interface is PCI 2.0 compliant and takes full advantage of the high bandwidth of the PCI bus by utilizing burst capability, write gathering and prefetching of data. The PCI-SCI bridge will generate configuration cycles using configuration mechanism #1. Both type 0 and type 1 configuration cycles will be supported to allow hierarchial PCI bus systems. The PCI-SCI bridge supports both memory space and I/O space. The use of memory mapped I/O is controlled by the host attachment node by using non-prefetch and non-write-gathered packets.

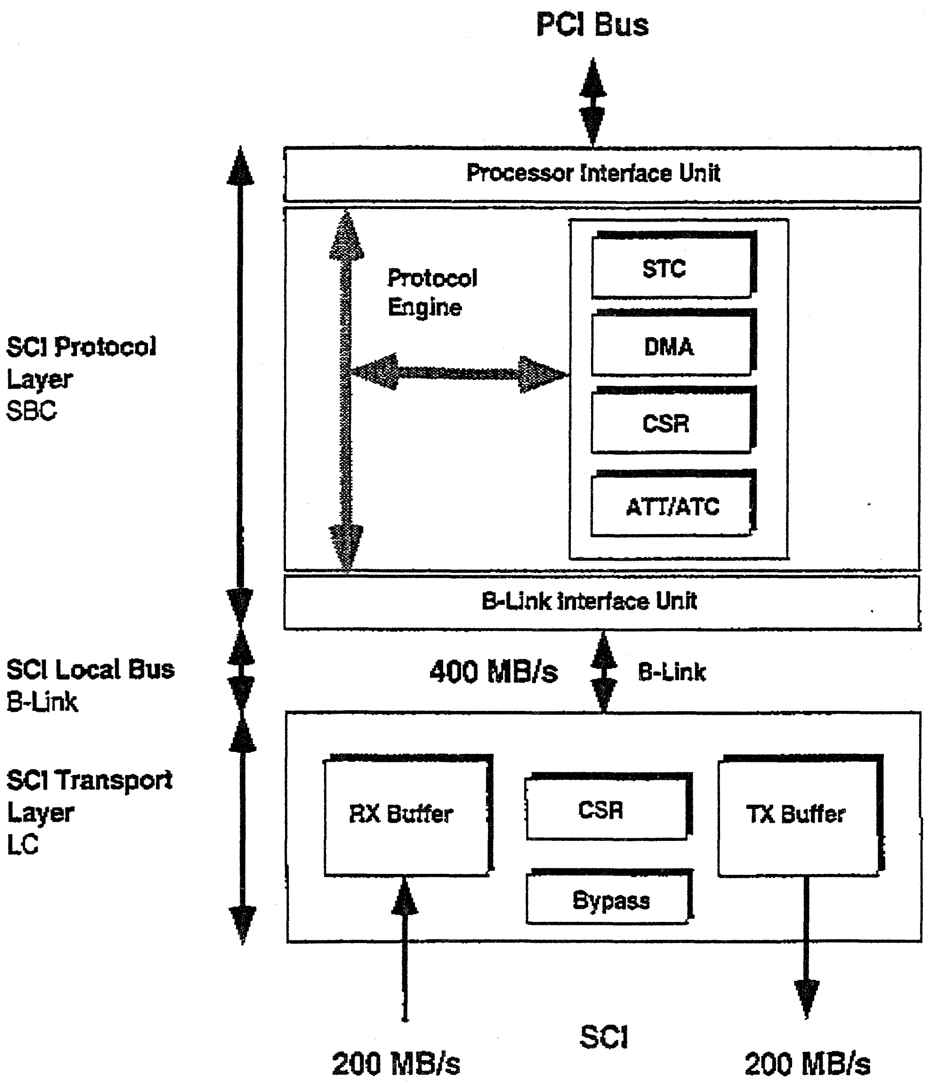
SCI Transport Layer Link Controller ("LC")

The Link Controller manages the SCI transport layer protocols. The LC has an input link, output link and a B-Link backside bus providing an implementation of the SCI transfer cloud. B-Link is a multimaster bus with support for up to 8 devices. The physical SCI interface operated at 100 MHz on a 16-bit parallel cable providing 1.6 Gbits/sec links.

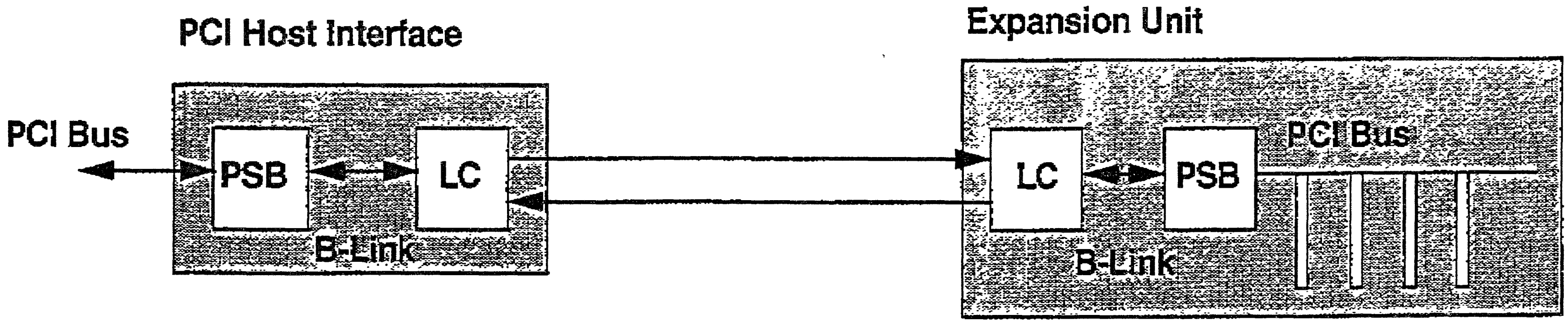
PCI Protocol Layer ("PtoB")

The PCI Protocol Layer is implemented in a custom ASIC, incorporating the PCI bus interface, a B-Link Interface Unit, and a microsequencer protocol engine. The PCI interface is controlled by the bridge using CSR (IEEE Std 1212-1991) access. The bridge includes an interrupt event reporting mechanism using standard CSR message or by issuing interrupts at the host attachment node.

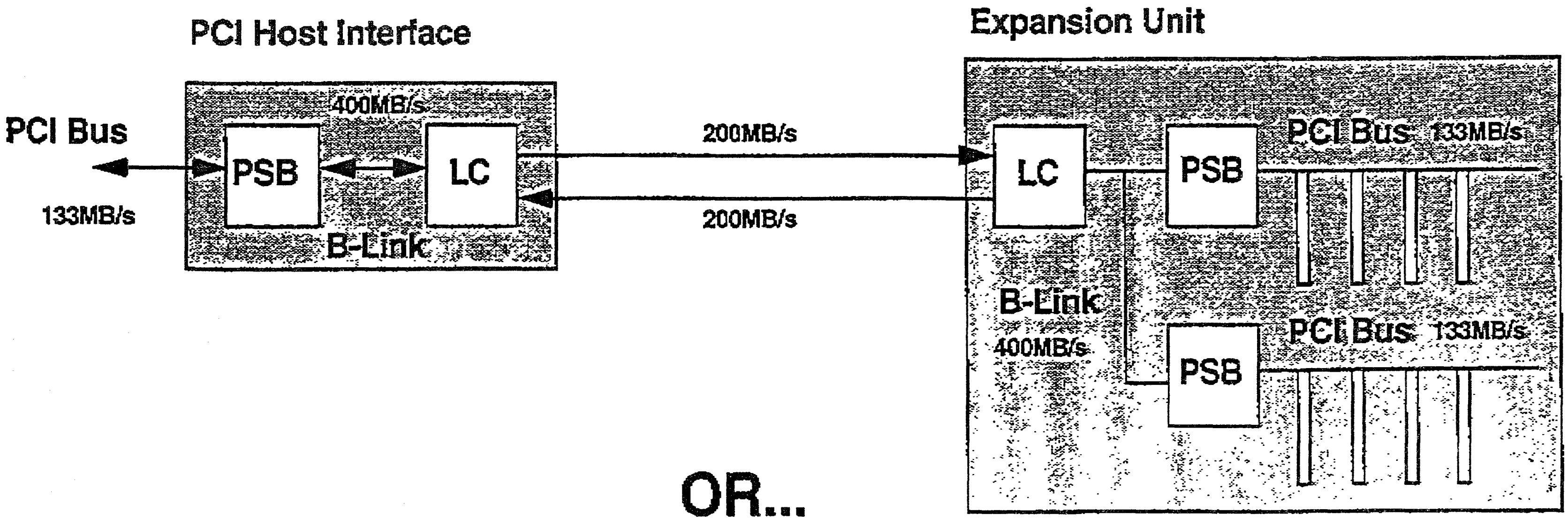
The PCI-SCI Bridge



Example configurations



OR



OR...

A PCI-SCI bridge for high rate Data Acquisition Architectures at LHC

H.Müller, A.Bogaerts, C.Fernandes, L.McCulloch, P.Werner

CERN Division ECP

CH1211 Geneva 23, Switzerland

Abstract

The RD24 project [Ref.1.] at CERN is constructing a demonstrator Data Acquisition System for the Large Hadron Collider (LHC) experiments [Ref.6.] using the Scalable Coherent Interface (SCI IEEE Std 1596) [Ref.8.] as interconnect. A large number of producer and consumer nodes may be equipped with PCI [Ref.5.] extensions and uniformly interconnected via a multi-node SCI system. The PCI-SCI bridge described here is an object of research for this application and therefore optimized for data movements away from the producers. Particularly important is therefore DMA and transparent memory access, as well as multi-cpu support. At a consumer side, this bridge should also allow for "dual ported" data flow, i.e. incoming and outgoing flow of data though separate PCI extensions of the same consumer CPU bus. This paper describes the design goals and tools chosen for a first PCI-SCI bridge in a CMC formfactor [Ref.2.] using the PMC physical layers [Ref.2.]. We use a 33 MHz PCI-PCI bridge chip on the PCI side and a 200 Mbyte/s SCI Link Controller on the SCI side.

1. Remote PCI local bus access over SCI

The reason for using SCI for Data Acquisition (DAQ) [Ref.3.] is SCI's capability of providing a very high bandwidth, uniform, bus-like interconnection between a large number of front end DAQ units (memories, embedded processors) and a large RISC processor farm [Figure 1]. The SCI network is implemented as a scalable, multistage system, consisting of SCI ringlets and switches. The PCI-SCI bridge is intended to equip DAQ units such as VME processor cards with PCI mezzanine provisions following the CMC and PMC proposed standard. Processor farm elements with PCI extension can be interconnected to the SCI network via the same type of PCI-SCI bridge. The çommunication between DAQ units and a CPU farm can be transparent over distance: remote PCI addresses can be mapped into the processor's address space providing transparent read or write operations over SCI. In addition, locked operations can be implemented at the bridge, allowing to implement robust and efficient synchronization over the SCI network without the need for additional control networks.

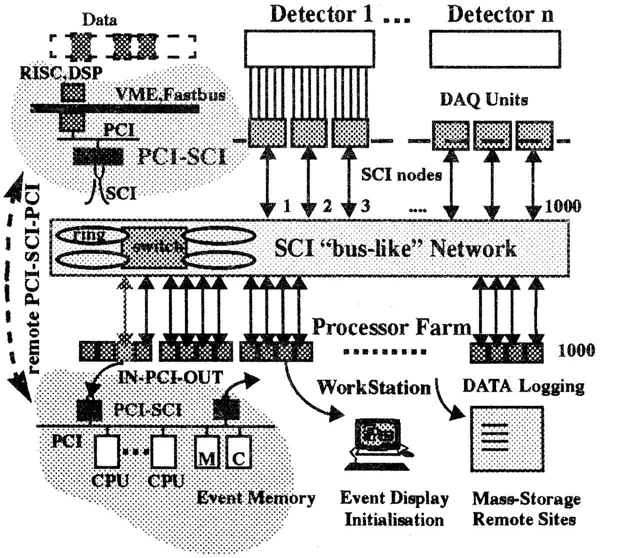


Figure 1: Remote PCI-PCI for large DAQ System

2. PCI-SCI bridge overview

The mezzanine card (CMC formfactor) of this PCI-SCI bridge [Figure 2] is targeted for integration in commercial VMEbus processor cards which are very popular in High Energy Physics DAQ systems. The same bridge logic can however easily be rerouted to be used within a PCI extension of a workstation.

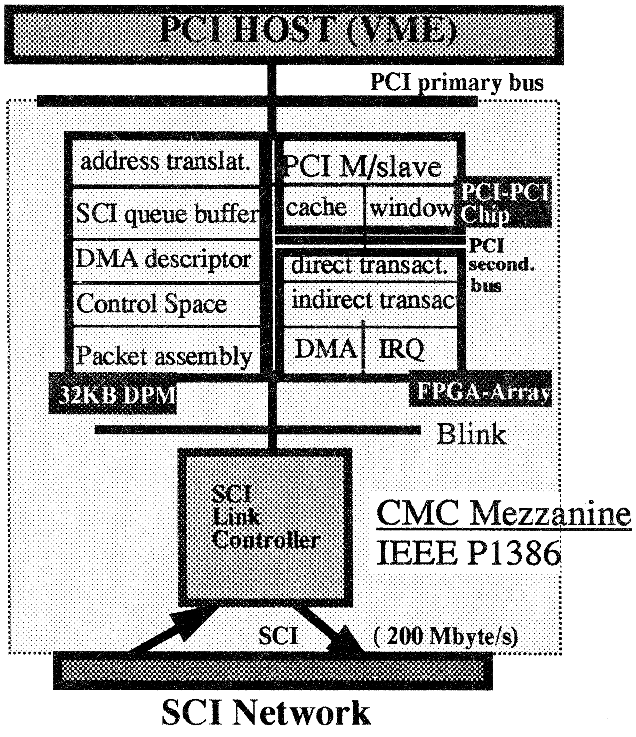


Figure 2: CMC Mezzanine card overview

For compliance with the PCI-PCI Bridge Architecture Specification [Ref. 4.] , a standard PCI bridge chip, the DEC21050 [Ref.14.] with pre-defined internal configuration space and configurable address windows is used. The hierarchy choice for this bridge specification is to allocate the primary PCI bus (needed to configure the bridge) on the PCI host side. The SCI node interface uses a Link Controller (LC) from Dolphin [Ref.7.]. This chip handles the SCI physical layers and a part of the logical layers of SCI. The user side of the LC is BLINK, a 64 bit packet synchronous bus [Ref.9.]. A fast dual port memory (DPM) is used as data transfer buffer between BLINK and PCI. It also serves various other purposes, such as address translation and queuing of SCI packets. The PCI-SCI specific bridge functions, such as direct or indirect PCI-SCI transactions, or DMA are implemented in an array of two ORCA FPGAs from AT&T [Ref.11.] which integrate each 15000 usable gates and more than 25000 user RAM bits.

3. Bridge logical overview

A variety of Functional Units (FUs) are distributed [Figure 3] over an array of two interconnected FPGA chips. FU's are developed as re-programmable logic, synthesized from VHDL or directly from a schematic capture editor. The scope of all FU's in this bridge is limited to interaction with VHDL models of the two buses on the bridge's internal periphery: the PCI-32 secondary bus of the PCI-PCI bridge chip and the 64 bit BLINK bus at the SCI chip.

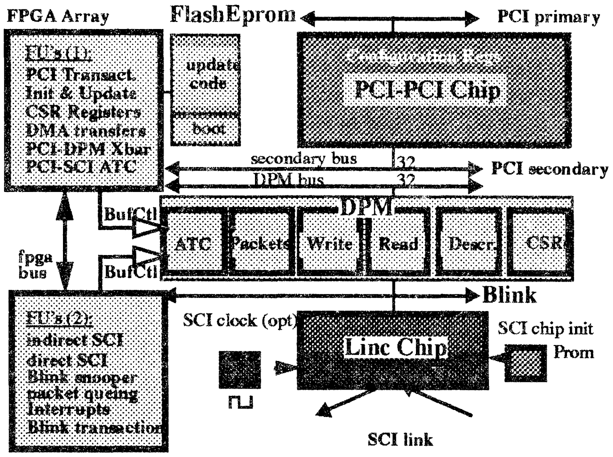


Figure 3: Logical Bridge Functions

The intermediate Dual Port Memory (DPM) is used by the FU's for various data buffering purposes. In addition, 15K bits of direct access storage for CSRs etc. are available within each FPGA. A default boot version of basic FUs is loaded at power-up from a read-only boot portion of the Flash Eprom. Update versions of FUs can consequently be loaded from the PCI host[1] into the writable portion of the Flash Eprom. A bridge specific CSR register is used to update and intitialize the functions.

3.1 Bridge access features

PCI memory and I/O transactions are supported. As part of the DECchip functionality, memory transactions can perform read ahead and write posting. These features are used to separate the cycle-by-cycle handshakes of PCI from the 16 or 64 byte packet transactions of SCI. The I/O transactions are reserved for accessing the bridge's internal CSRs or single byte/word reads to sensitive byte addresses without causing side effects.

4. Bridge physical overview

Data and control flow in this bridge architecture [Figure 4] are orthogonal: data is transferred vertically between the PCI32 bus and the 64 bit BLINK bus across the DPM and the byte crossbar (XBAR inside FPGA1). Control information is passed horizontally between the FPGA array and the DPM, the Link Chip and the PCI bridge chip.

The Physical Units (PUs) have been minimized to fit on both sides of a single-width CMC mezzanine card and allow for a maximum degree of flexibility and re-programmability. The 32 bit PCI data bus is connected to BLINK across the DPM. PCI data are transferred between 2*32 bit DPM ports on the PCI side and a 64 bit wide port to BLINK. Initially, the BLINK clock is 1/2 of the nominal 33 MHz PCI clock. On the BLINK side, the DPM is always addressed as a synchronous 64 bit port, providing complete SCI packets in the format and timings required by BLINK. On the PCI side, a more asynchronous mode via WAIT stated data cycles is allowed. The secondary PCI bus and the DPM bus are connected via a 4 byte crossbar (XBAR) function for byte reordering.

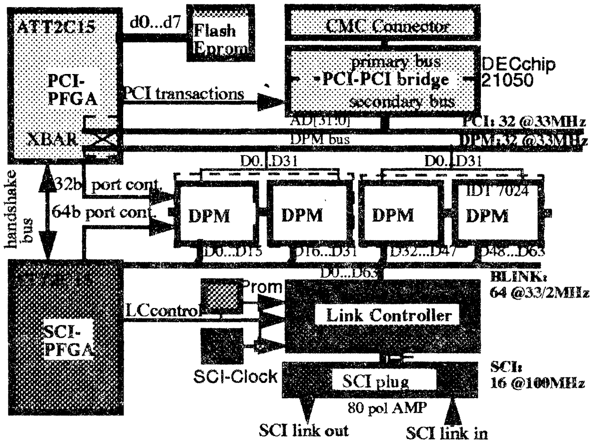


Figure 4: Physical Bridge Functions

4.1 SCI Link Chip

This new Dolphin SCI Link chip L5A4241 [Ref.7.] comes in a 208 pin MQUAD package and handles the SCI link level at 200 Mbyte/s. To the application side it provides the 64 bit BLINK interface with big endian byte ordering. The CMOS chip requires 5 Volts and consumes less than 3 Watts. The LC decouples the SCI link domain (differential PECL) from the BLINK (TTL) domain, allowing for independent clocks. The SCI link protocols and two primary packet queues are available together with 9 CSR registers. The configuration CSR is loaded at power up via a serial PROM. The LC supports multiple outstanding requests and outstanding subactions of SCI, which need to be queued via additional DPM resident buffers.

4.2 PCI-PCI bridge chip

The DECchip 21050 [Ref.14.]implements 2.0 PCI compliant signal drivers and works with a clock frequency of up to 33 Mhz. It is implemented in a 208 pin PQFP package and consumes roughly 1.5 Watts. The 21050 provides concurrent primary and secondary bus operation for bidirectional transaction forwarding, i.e it can act both as a target or initiator on both of its sides. Memory transactions can be filtered through two programmable memory address regions, one of which is prefetchable making use of an internal 256 byte cache.The cacheline register allows defining up to 256 bytes of cacheline, of which the values 16 or 64 are useful for SCI packet generation.Write posting of up to 8 DWORDS is available.

5. Dual Port Memory

The central component of the SCI-PCI bridge is the dual-ported RAM. It consists of four IDT 7024 chips in TQFP 100 package which are organized as 16 * 4K to give a total of 32 Kbyte, upgradable via double density IDT 70261 chips. The PCI access to the DPM is via the I/O and the non-prefetchable memory space. The DPM is partitioned as described in the following sections:

[1] we use a XMVME-1600 PowerPC card from Motorola

5.1 Packet queue buffer

The implementation of multiple outstanding SCI transactions allows to make efficient use of SCI bandwidth. Response latencies on PCI can be made use of by queuing request or send packets before eventually generating SCI retries. Both outgoing and incoming queues are added to the packet queues built into the LC.

5.2 DPM transfer buffer

The asynchronous DPM supports direct data pass-through between the asynchronous PCI and synchronous BLINK, if simultaneous access to the same address is avoided via an external comparator logic. However, since the transfer buffer is seen from the PCI side as two subsequent 32 bit words and as one 64 bit word on the BLINK side, a maximum speed for such unbuffered transfer is one half of the PCI clock on the BLINK side. Whilst 33 MHz operation on BLINK with intermediate packet storage is possible, we chose to start with a simplified clocking at BLINK at 1/2 of the PCI clock.

5.3 PCI-SCI address translation lookup table

The address translation table for transparently mapped SCI addresses is part of the DPM and uses 1024 words (32 bit) to look up 1K SCI node addresses [Figure 5]. The translation is needed to extend the 32 bit address space of PCI to a physical 64 bit SCI space.

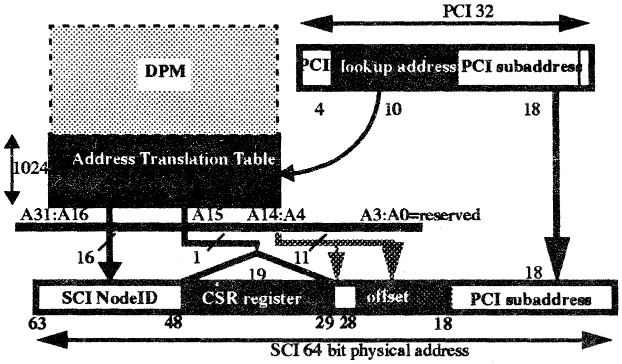


Figure 5: Address Translation PCI->SCI

Ten PCI address bits within the PCI slot range are used as an index into the 1024 word deep lookup table of the DPM. Four reserved output bits A3:A0 are available. Together with the directly connected lower 18 bits of the primary PCI subaddress the 28 bits from the address table output are used to form the physical SCI target address. The 19 bit wide SCI address field SCI<47:29> is generated from the bit A15 from the address translation table. This field distinguishes between SCI memory and the CSR space, A14 between private and register space.

5.4 Semaphore space

In order to provide a uniquely resolvable multi-CPU access to all DPM data structures, each 32 bit DPM word has an associated semaphore bit mapped out from the bottom part of the DPM [Figure 6], which can be derived using a simple shift arithmetic on the word address. This scheme requires a dedication of 1/32 th of the DPM for semaphores.

5.5 Descriptor space

Descriptors for the DMA engine and for SCI packet mode operation are stored within the DPM. Special descriptors are the SCI transaction identifiers for outstanding requests (32 max) which are used as pointers to the request and response queue buffers.

5.6 PCI-SCI Configuration space

The jumperless standard 256 byte configuration space of the PCI-SCI bridge is allocated within the DPM and gets preloaded at power up from the Flash Eprom. On the primary PCI side, the DECchip has its own configuration space.

6. Locked operations

Indivisible operations need to be supported by the bridge in a simple way to allow both PCI or SCI resident CPUs perform resource allocation. This requires implementation of binary semaphores for the DPM, generation of SCI lock primitives via the packet mode and generation of PCI locks for incoming SCI locked transactions to PCI.

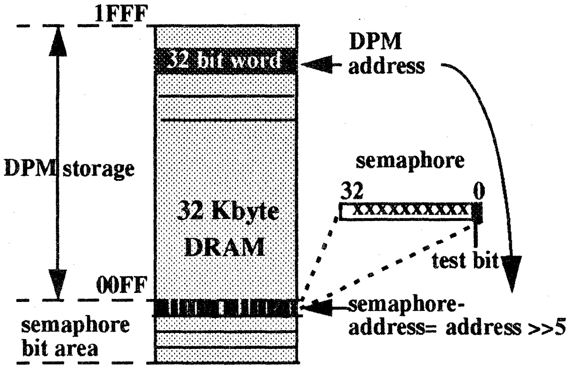


Figure 6: Semaphore bit addresses in DPM

6.1 DPM semaphore space reads and writes

For resource sharing of the DPM, the access to semaphore space of the DPM from both PCI or SCI is converted as follows: A read to a semaphore address is converted by the bridge into a read&set of the semaphore bit before the external read cycle completes. Any write cycle to a semaphore address is converted to clear of the semaphore bit.

6.2 PCI LOCK transactions

The PCI LOCK protocol is available from SCI to PCI address space conditioned by initialisation of PCI memory or I/O space access. Incoming SCI locked transactions are converted to PCI locked transactions. The DECchip generates and transmits lock requests via its LOCK signals on both its primary and secondary buses. Resource locks generated from the PCI host are transmitted and possible for the DPM spaces only.

7. Chain mode DMA

A DMA engine allows direct memory transfers between PCI and SCI memory under control of an external DMA host. The engine is controlled via chained, DPM-resident descriptor blocks which are treated in sequence. Multiple DMA requests from different CPUs are handled via a first-come-first-served access to a DMA control register. This CSR stores pointers to the head of a chain descriptor like a FiFo.

7.1 DMA Descriptor Block fields and functions

Very similar to the DMA descriptor of the Apple SCI transparent interface[Ref.15.] our descriptor block [Figure 7] for one contiguous block of data within a DMA chain consists of four consecutive 32 bit words describing both local PCI and remote SCI addresses (source and destination). The descriptor can be set by both PCI or SCI. The command field contains a DMA command, the IRQ field contains interrupt options and the Xfer field provides size and current count of the transfer. The XT bit allows to enlarge the size of the descriptor block by a factor of 2 for additional storage of 64 bit PCI addresses. The two ST bits provide status information on completion of this descriptor. Descriptors are always initialized with ST=10. On completion of the DMA the upper bit is cleared, the other bit serving as an error status. The IRQ field contains four bits for generating interrupts on completion of the DMA or on errors. The Xfer field indicates the number of bytes to transfer (1 - 1MB). A zero in this field indicates a skip to the next descriptor block in memory.This field must always be readable to provide status information. The BW field allows to reduce the DMA transfer speed to prevent contention at the receiving node. The 3 bit command field initially implements 4 commands: Write, Move, Read and STOP.

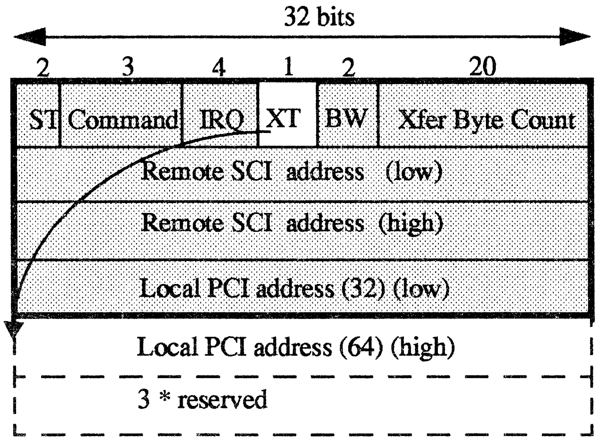

32 bits
2
3
4
1
2
20
ST
Command
IRQ
XT
BW
Xfer Byte Count
Remote SCI address (low)
Remote SCI address (high)
Local PCI address (32) (low)
Local PCI address (64) (high)
3 * reserved

Figure 7: Chain DMA descriptor block

8. Transparent PCI-SCI transactions

PCI address windows of this bridge can be mapped into SCI. Such transparent access is the key feature for implementing remote memory access (example PCI-SCI-PCI) or shared memory applications. In this bridge, the prefetchable memory space is mapped into SCI transactions the I/O space and the non-prefetchable space is only used to access the local DPM and CSRs [Figure 8]..

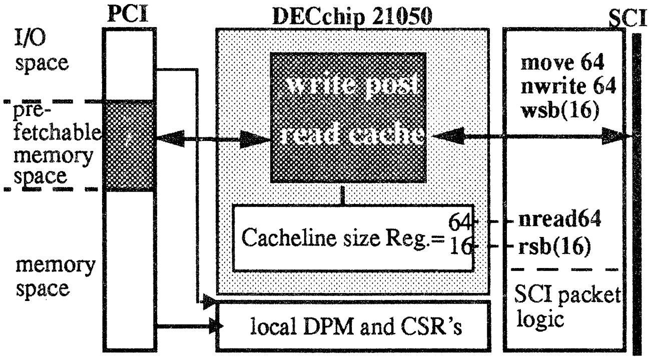


Figure 8: PCI Space to SCI transactions

8.1 PCI I/O space mapping

The I/O space is local to the DPM and CSR registers of the bridge. It allows access to individual bytes without side effects.

8.2 Incoming SCI requests

Requests from SCI are forwarded to PCI if prefetchable memory, as defined via the DECchip, is addressed, otherwise these requests are refused. The DPM and CSRs of the bridge are accessible within the CSR register space of the bridge.

8.3 Prefetchable memory space mapping

The DECchip provides the possibility to define prefetchable memory windows to be transmitted through the bridge. This memory space uses a 256 byte cache whose cacheline size can be defined via the cacheline register. For transparent SCI read transactions, a cacheline of 16 byte can be used to generate the 16 byte rsb transactions, or a cacheline size of 64 byte can be used to generate 64 byte nread64 transactions. Similarly for PCI->SCI writes, write posting is a feature of the DECchip allowing to post up to 8 DWORDS (32 byte) without need for a response. This feature is used to build byte directly mapped SCI write transactions wsb, nwrite64 or move64.

9. Packet mode transactions

All SCI transactions supported by the LC, in particular SCI locked transactions can be implemented as packet mode transactions with minimal hardware overhead and more software overhead. In this mode, SCI packets are precompiled into packet mode buffers which are reserved in the DPM at fixed addresses, each associated with a descriptor and an error summary word in the CSR space. The procedure to send a packet mode request and eventually read the result is as follows:

Allocate a packet mode buffer in case of multiple processors accessing the SCI-PCI bridge from PCI and compile the complete packet in the packet mode buffer. Update the descriptor to indicate the request sent - no response status. This transaction triggers also the transmission of the request packet. Poll on the descriptor and upon arrival of the response packet, assert the response (and if required the error) bits together with the packet number.Determine the address of the response packet using the packet number and read out the result, clear the response buffer.

10. Packet queues

An SCI node requires four buffer queues: one pair for output request and response, and one pair for input request and response. The LC has only very limited buffer space, but allows an unlimited number of external buffers. For simplicity, we use the SCI transaction ID (which may count up to 64 for uncompleted transactions) as pointer to the request-output and response-input queues, however we only allow up to 16 buffers for both incoming or outgoing packets. The 64 buffers of fixed length of 128 bytes require roughly 8 Kbyte of the DPM space.

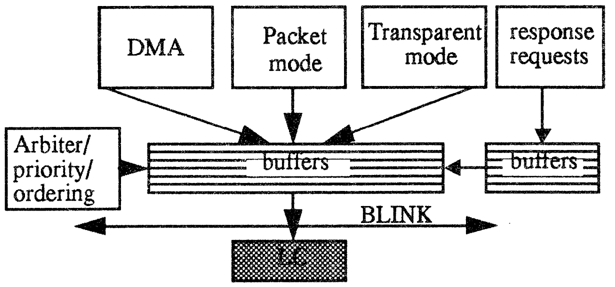


Figure 9: Outgoing queues to Blink

10.1 Outgoing queues

There are three sources for outgoing requests: DMA, transparent mode packets and packet mode [Figure 9]. These must be merged with the response requests. All packet sources store their requests in the output request queue buffers. An arbiter logic requests BLINK access according to a desired priority distribution. The same logic may be extended to manage packet ordering.

11. FPGA design environment

The development of functional units for the ORCA [Ref.11.]FPGA is based on either VHDL using an editor or alternatively schematic capture using CADENCE CONCEPT [Ref.12.]. A VHDL source file is compiled and simulated via LEAPFROG [Ref.12.]. Once functionally correct, the VHDL model is passed through the EXEMPLAR [Ref.13.] sythesizer tool to create an output file in XNF format. The internal mapping of a target FPGA structure is performed via the NEOCAD MAPSH [Ref.10.] tool. The result is passed to the router which generates a database for an interactive editor, EPIC, allowing to look at routing and to edit it if necessary.

Timing details and critical path delays can be optionally specified as well as separate nets for slow and fast timings. The TRACE tool allows for static timing analysis and it gives information about the maximum achieved frequencies and combinatorial delays.

The routed output is returned to the mapper for back annotation and then can be post simulated in CADENCE.

The design cycle may need reiteration in its source code. Once complete, a bitstream for a PROM programmer is generated which allows loading a Flash Eprom with the FPGA boot configuration

11.1 Partitioning into FPGA arrays

The PRISM tool will be used[1] for partitioning the design block into two FPGAs. In a first partitioning, a quick fit is created. This is edited and the design is repartitioned following a guide file. The partitions are re-routed and then analyzed with EPIC.

12. CSR and Configuration Space

There are three CSR modules: the PCI configuration space and the CSRs within the Link Controller and bridge specific CSRs to be implemented in programmable logic. The SCI CSR space is implemented according to the CSR IEEE-1211 architecture. All CSRs of the PCI-SCI bridge can be completely configured from PCI alone after a reset. The configuration of the PCI-PCI bridge can only be performed via the PCI host. Both the CSRs of the Link Controller and the bridge specific CSRs can always be accessed via SCI.

13. Acknowledgements

This work is funded via the DRDC (now LCRB Computing Review Board for LHC) committee at CERN and uses much infrastructure of the CERN ECP Division. We thank Dolphin SCI Technology, OSLO Norway and Apple Computer ATG in Cupertino USA for many discussions. Several concepts are derived from design ideas from V.Lindenstruth, LBL Berkeley for the SCI-PCI bridge of the STAR experiment. The SCI-PCI bridge for STAR is conceived along the same principles however does not use the DECchip.

14. Information

RD24 maintains an anonymous ftp server rd24.cern.ch and is on WWW: http://www1.cern.ch/RD24

The detailed design specification is available WWW, protected via CERN copyright. Commercial enquiries should be sent to the CERN Finance Division, CH1211 Geneva 23, Switzerland.

15. Bibliography

1. *RD24 Collaboration,* **"RD24 Status Report, Application of the Scalable Coherent Interface to Data Acquisition at LHC"**, CERN/DRDC 93-20, 5 May 1993
2. **Draft Standard for a Common Mezzanine Card Family:** CMC P1386/Draft 1.6 and **Draft Standard Physical and Environmental Layers for PCI Mezzanine cards: PMC, P1386.1, Draft 1.6** *IEEE Standards Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA*
3. *H.Müller* **"Scalable Coherent Interface Applications to DAQ"**, Overview talk at the DAQ Conference Fermilab Oct.26-28. (ftp rd24.cern.ch sci/Talks+Papers/FNALtalk.ps.)
4. **PCI to PCI Bridge Architecture Specification,** *Rev 1.0 PCI Special Interest Group, P.O. Box 14070, Portland, OR 97214, USA*
5. **PCI Local Bus Specification, "Review Draft"** Revision 2.1 October 1994, *PCI Special Interest Group, P.O. Box 14070, Portland, OR 97214, USA*
6. **Information on LHC and: ATLAS,CMS and ALICE information and technical proposals are accessible via the World WideWeb:** http://www.cern.ch/ under "Activities"
7. **Link Controller Target Specification DIS656A**, *Dolphin Interconnect Solutions AS, P.O. Box 52, Bogerud, N-0621 OSLO, Norway*
8. **Scalable Coherent Interface IEEE 1596-1992**, Distribution and copyright by *IEEE, 345 East 47th Street, New York, NY 10017-2394, USA*
9. **Backside Link (B-Link) for Scalable Coherent Interface (SCI) Nodes**, Draft 2.3. *Dolphin Interconnect Solutions AS, P.O. Box 52, Bogerud, N-0621 OSLO, Norway*
10. **FPGA Foundry: Device Independent Modular Toolset for FPGA Design,** *NeoCAD Inc. 2585 Central Avenue, Boulder, CO 80301*
11. **Optimized Reconfigurable Cell Array (ORCA)**, *AT&T Microelectronics, Dept-500404200, 555 Union Blvd, Allentown, PA 18103, USA*
12. **CONCEPT, LEAPFROG are tradenames of:** *Cadence Design Systems, Inc. 2 Lowell Research Center Drive, Lowell, MA 01852-4995*
13. **EXEMPLAR** is a tradename of Exemplar Logic, Inc. 2550 Ninth Street, Suite 102, Berkeley , CA 94710, USA
14. **DECchip 21050 PCI-to-PCI Bridge**, Digital Equipment Corporation, Maynard Massachusetts, USA
15. **Implementation Issues of Bridging SCI to a conventional Processor Bus,** G.Stone & D.North of Apple Computer ATG in: *Proceeding of The First international Workshop on SCI-based High Performance and Low-Cost Computing, August 17-18, 1994 St.Clara University*

[1] Currently this is not yet available for the ORCA FPGAs

ACCESS.bus™: The Missing Communications Link For Easy, Inexpensive Connectivity Between the Host System and Computer Peripheral Devices

by Richard J. Fisher, Applications Manager
Microchip Technology Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224
Ph. (602) 786-7200 Fax (602) 917-4005

The ACCESS.bus™ is the new standard for device communication and connectivity developed by Digital Equipment Corp. and Philips Semiconductor. These two companies released this technology publicly in order to make it an industry standard. More than 100 industry leaders, including Philips, NEC, Fujitsu and Microchip, created the ACCESS.bus Industry Group (ABIG) to support the implementation of this standard.

ACCESS.bus provides for the first time a two-way communications channel through which both external peripherals and other "intelligent devices," such as the video board, power supply and power management devices (System Management Bus), can exchange data with the host system. This feature enables end-users to control peripheral and device attributes, like monitor resolution, mouse sensitivity and power management modes, through a single easy-to-use software interface.

The new specification also allows up to 125 peripherals and input devices, such as monitors, keyboards, mice, bar code scanners, etc., to be connected to any type of system through a single port at any time, in any order--without the need to exit applications and reboot the system. This makes peripheral connection easy. No longer will users have to figure out which cable goes into what slot in the back of the computer--or how to hide the ubiquitous wire maze. Just plug in a device and it will work.

The industry standard I^{2}C™ (Inter Integrated Circuit) protocol is used for all messaging. In short, ACCESS.bus is relatively simple to implement physically, and thanks to I^{2}C, can be used with hundreds of available microcontrollers, memory and other components. This translates into low cost and therefore potential widespread usage.

As intelligent devices are also sharing the same system interface, system management functions and peripheral communications can be performed effectively with fewer interrupt lines and input/output addresses. Further, since a large number of system tasks are delegated to the "intelligent" devices, the CPU is interrupted less often and can therefore get more important tasks, like running applications, completed more quickly and efficiently. The new bus utilizes relatively inexpensive silicon compared with what is needed by, for example, high-speed devices, and fewer components than are currently being used. And, as OEMs and system designers need to implement only one communications channel, system development cost and time is considerably reduced.

Because ACCESS.bus is a "Plug & Play" standard and thus automatically configures both system and peripheral devices, installation is simple. Almost limitless easy upgrade potential is also achieved.

What ACCESS.bus provides, in short, is the missing communications link. By itself it serves all low speed peripherals. But it can also expand to offer a two-way communication port through which other buses and their dependent devices can communicate to the host system and, ultimately, to the end user. ACCESS.bus is included in the most sophisticated layer of the Display Data Channel™, the standard created by VESA™ for transmitting configuration information from a video monitor to a host computer, and the System Management Bus, the communication specification developed by Intel and Duracell for on-board system management devices.

And ACCESS.bus will continue to expand.

Editor's Note: Mr. Fisher is treasurer of the ACCESS.bus Industry Group.

The Microchip name and logo are registered trademarks of Microchip Technology Inc. ACCESS.bus is a trademark of the ACCESS.bus Industry Group. VESA is a registered trademark and DDC is a trademark of Video Electronics Standards Association. I^2C is a trademark of Philips Semiconductor. All other trademarks are the property of their respective owners.

Access Bus Industry Group (ABIG), 370 Altair Way, Ste. 215, Sunnyvale, CA 94086, (408) 991-3517/3773 (Fax)

S^2I: The Key to Direct-Attach PCI Disk Storage

Martin Freeman

Philips Research Palo Alto
4005 Miranda Ave, Suite 175
Palo Alto, CA 94304

(415) 354-0329; martin@prpa.philips.com

Abstract

Advances in storage and high-performance interconnect technologies are offering opportunities to approach the next step in storage interface evolution. Opportunities exist for integrating the storage interface into the processor memory hierarchy to achieve high-performance at low cost. For example, in today's high-performance systems, host adapter cards are used to interface disk drives to the system I/O bus; better storage interface integration can eliminate these adapter cards. Furthermore, recent advances in storage controller design can streamline the storage interface operation and help produce the lowest cost device interface.

The emerging draft IEEE P1285 Scalable Storage Interface (S^2I) standard addresses these opportunities in providing a low-cost, high-performance storage interface that is simple, scalable, RAID and multiprocessor friendly and interconnect independent, i.e. any number of physical levels can be used including PCI, SCI, RamLink, SRAM, DRAM, etc. For the PCI physical level, S^2I/PCI disk drives are possible that plug directly into the PCI socket on PC motherboards.

Economonies of scale in the PC marketplace should drive the cost of S^2I/PCI disk drives down to competitive levels. Because disk drives just plug into the PCI backplane, they can be aggregated to form RAID subsystems. Since PCI bus throughput is 132 Mbytes/sec for 32 bit systems and 264 Mbytes/sec for 64 bit systems, such subsystems are ideal for database servers and video servers. Using S^2I with SCI, such systems can be joined together to achieve an interface throughput of 1 Gbyte/sec.

S^2I provides an interface hierarchy of two levels: beta and gamma. The beta level is optimized for 32 bit uniprocessor systems, while the gamma level is optimized for 64 bit multiprocessor systems. The interface hierarchy exposes the the internal disk drive device buffer to direct access by the system. It takes advantage of disk drives having support for on-the-fly error correction by providing efficient mechanisms for accessing device-local buffers. These buffers are managed by system software. To support data movement, S^2I disk drives can also execute commands from system memory (or the drive's internal buffer). Most of these commands transfer data between the buffer, the media, and the system memory spaces. Other commands are available to provide branching capabilities and to synchronize command execution.

A PCI-Based Industrial Backplane

PICMG Consortium
301 Edgewater Place, Suite 220
Wakefield, MA 01880

The PCI Industrial Computer Manufacturers Group (PICMG) is a consortium of vendors who are designing a specification for PCI-based systems and boards for use in industrial computing applications. The consortium's mission is to extend the PCI standard, as approved by the PCI Special Interest Group (PCI SIG), to incorporate industrial single board computer systems.

PICMG intends to offer industrial equipment vendors a common specification, thereby increasing the availability and reducing the costs of industrial PCI standard-based products. The PCI industrial standard will provide a clear upgrade path for OEMs wishing to migrate to it.

PICMG's session will highlight and discuss the design challenges involved in porting the PCI standard to passive backplane and other bus structures commonly used in the industrial marketplace. Topics will include how to further develop the existing PCIMG standard, selecting chips and BIOSes, compliance testing, and current and future product availability.

The current standard will give industrial users all the features of the PCI bus, while maintaining access to the widely available ISA bus cards for purposes such as data acquisition. Vendors presently provide CPU cards and backplanes compliant with the specification.

For more information, please contact the chair, Pierre McMaster, at the following address:

Pierre McMaster
Chair, PICMG
Teknor Microsystems, Inc.
616 Cure Boivin
Boisbriand, Quebec, Canada J7G 2A7
(800) 387-4222
(514) 437-5682/8053 (fax)

QuickRing™: A Low-Cost, High-Speed Seamless Interconnect

Sean Long
Product Launch Manager
National Semiconductor
Mail Stop A1545
PO Box 58090
2900 Semiconductor Dr.
Santa Clara, CA 95052
(408) 721-3046/737-7218 (fax)
seanl@joyride.nsc.com

Sean Long graduated with a BSC Honours degree in Electrical and Electronic Engineering from Aston University, Birmingham, England. He is a member of the Institute of Electrical Engineers (IEE). After graduating, he worked as a design engineer for Schlumberger and a design consultancy, designing a variety of microprocessor and Digital Signal Processing based systems. From 1988-1992 he worked for National in Europe, doing Technical Marketing covering programmable logic and memory products. From 1993 to the present he has worked for National, based in Santa Clara, as a Product Launch Manager with responsibility for QuickRing.

National Semiconductor™

National Semiconductor QuickRing™ Interface Forum

For more information contact:
Sean Long
Product Launch Manager
408.721.3046

As our industry demands faster speeds and greater interconnectivity, semiconductor technology becomes even more critical. At National Semiconductor, we are in the business of creating *Technologies for Moving and Shaping Information™*. Our semiconductor technologies for data transmission and computer I/O set global standards.

Moving functions are those that transport data, internally via bus architectures, or externally, between computers or systems. QuickRing™ is unique in its ability to move streams of data at extremely high speeds, both for in-box and box-to-box applications. In fact, QuickRing's ability to solve the I/O bus and interconnect bottleneck is one great example of how National is moving and shaping information to benefit our customers.

The National QuickRing datastream controller is composed of an architecture, a physical layer hardware specification, and a communications protocol. It is used for high performance data streaming between devices, cards, or systems. The National QuickRing can replace buses or work in complement with them.

The National QuickRing controller gets around the I/O bottleneck with a point-to-point interconnect scheme. Unlike multi-drop buses which are limited to one transaction at a time, the controller's architecture allows for concurrent data transfers. The National QuickRing can support between 2-16 nodes in a single ring; plus a ring of rings structure, each capable of sending and receiving data simultaneously.

Performance between any 2 nodes in a point-to-point scheme using the National QuickRing is up to 200 MegaBytes/sec. For a 16 node ring, an aggregate bandwidth of up to 1.7 GigaBytes/second is achievable.

The National QuickRing is a low cost, high-speed interconnect which allows seamless interconnects within and between systems. With National QuickRing, moving large amounts of data in high-speed LAN routers and hubs, servers, and embedded computing, is not a problem.

System designers can use this technology to enhance the performance of current products or design new ones. For further information on the National QuickRing, contact:

General Product Information
Tel: 800.272.9959

Application Hotline
Tel: 408.721.5457

Internet

For individual questions, send to:
quickr@tevm2.nsc.com

On Line Bulletin Board
For latest product status:
Tel: 408.721.2542 (8-n-1)
2400-14400 bps

National Semiconductor™

For open forum participation
(reflector) send to:
qrlink@lightning.nsc.com

The National QuickRing Design Handbook

Includes application notes and data sheets for the QR0001 and QR1001. To order, call 800.272.9959
and ask for lit # 550067

RACEway: A Scaleable Interface for Real-Time Multiprocessor Systems

Barry S. Isenstein
Director, Strategic Marketing
Mercury Computer Systems, Inc.
199 Riverneck Rd.
Chelmsford, MA 01824-2820
(508) 256-1300/3599 (fax)
isenstein@mc.com

Barry Isenstein is responsible for product planning, message development and strategic planning at Mercury. Before joining the company in 1984, he was senior project scientist at Coulter Biomedical Research Corporation, a manufacturer of digital image based analysis systems. There he was responsible for design and implementation of image processing algorithms for automated cytology. Previously, he spent three years on staff at Case Western Reserve University at the Picture Processing Laboratory facility of the Biomedical Engineering department. There he worked on image processing and pattern recognition projects. Isenstein earned a master's and bachelor's degree in biomedical engineering from Case Western Reserve University.

PCI'95 Conference

March 30-31, 1995

Santa Clara, California

BEHAVIORAL VALIDATION AND ITS APPLICATION TO PENTIUM CLASS PROCESSORS

Mark Scheitrum and Alan Smith
CPU Technology, Inc.
47212 Mission Falls Court
Fremont, CA 94539

ABSTRACT

Error-free operation of complex computers and systems is an area of critical concern to the entire computer industry. Current validation approaches by the manufacturers do not adequately represent the interests of systems integrators and end-users. The entire industry must participate in establishing measurable standards for compatibility and enforcing these standards consistently. This paper describes an improvement in processor and computer validation that is achieved through specification-based behavioral validation. Specific benefits that are achieved by the manufacturer are detailed. Benefits to the systems vendors and end-users are also presented. The Pentium class processor validation suite is summarized. The generality of this approach and application to software and other systems logic is discussed.

BACKGROUND

Even as computers are being trusted with more critical responsibilities, they are becoming too complex to trust. Each generation of computers is exponentially more complex than the preceding generation. Computers are critically involved in flying our planes, monitoring our health, driving our cars and controlling our finances. The entire population relies on computers to have a high level of compatibility, reliability and upgradability. Traditional methods of verification, which rely on manual, ad-hoc testing and screening with popular applications software, are inadequate to guarantee that all functions of a processor are actually working properly. Validation responsibility has been left to the designers exclusively, without appropriate tools, without industry participation, and without measurable and enforceable standards. This approach forces the end-user into an unwanted key role in the debug process. The entire industry assumes a tremendous risk and liability as a result. Manufacturers, integrators, and users must participate responsibly by setting and enforcing more formal compatibility and upgradability standards, especially where life and property are at stake.

COMPLEXITY AND THE MYTH OF THE BUG FREE COMPUTER

It must be restated that a computer is an extremely complex, highly precise device. The value of a computer comes from the fact that it can perform a range of complicated instructions, absolutely correctly at an extremely high rate. To perform and validate all combinations of possible instructions, modes, addresses and data patterns would take longer than the lifetime of the sun for even a simple processor. In something this complex, perfection is unverifiable and the concept of 'bug-free' is a myth.

This combination of complexity and uncertainty is nothing new. We deal with it daily in many other natural and man-made systems. But computers and other electronic systems are so new and so capable that by the standards of other industries, they seem perfect. It is only when measured by their own standards that they appear flawed. In what other technology would a calculation error of less than one in 10,000 every 1,000,000,000 events be considered anything but perfection?

It is up to the entire industry to establish the standards of compatibility for all computer systems and to ensure measurement of these standards.

VALIDATION IS A SHARED RESPONSIBILITY

Designers, manufacturers, integrators and users have different validation objectives. Of course, each of these groups wants the final device to have no bugs, but their direct involvement is limited.

Designers need to verify each design element and get the processor to a level of functionality to prove performance assumptions as quickly as possible, so they can move on to the next design.

Manufacturers need to extend the architecture to open up new applications and to ensure compatibility to some acceptable criteria as quickly as possible so that the product can be shipped.

Systems companies and device integrators want consistent testing applied by all manufacturers to provide verifiably compatible devices against standard criteria.

End-users want as much testing done at every level as is possible. They also want low price, high performance, new features and no problems with their applications, even future ones. (They want everything!)

In a situation where a processor cannot be tested completely, only those groups who are involved in the validation process can be assured of having their interests represented.

INDUSTRY COMPATIBILITY TESTING GOALS

For any processor, it is important to ensure that the validation approach has the following characteristics:

1. It is based on a measurable and enforceable specification. (Claiming compatibility to an operating environment is not measurable.)
2. It provides comprehensive specification coverage. It tests the defined capabilities of the processor, not just the ones in common usage.
3. It is applicable throughout the development process as well as on the finished device. It minimizes the number of 'flaws' in the design rather than find them in the finished product.
4. It is available to all manufacturers of compatible processors, to ensure that all manufacturers are measured to the same consistent standard.
5. It integrates new tests as the specification evolves or as the industry requires.
6. It is an efficient diagnostic and reporting tool, which directly identifies any problems that it detects.
7. It provides repeatable and predictable test results.

BEHAVIORAL VERIFICATION TECHNOLOGY™

Behavioral Verification Technology™ (BVT) was invented to address the validation interests of all of the participants in the computer industry. It enables the rapid development of formal instruction-set architecture validation suites for any computer. Among the areas where BVT already provides value are commercial computing (x86 instruction-set) and avionics computing. Every computer intended to last for more than one generation will benefit from a BVT-based validation suite.

The ability to properly validate a processor's defined functions as early as possible in the design cycle is the major problem faced by processor manufacturers. A method is needed which automates the process of validating processors to remove design flaws before the customers see them.

Behavioral Verification Technology provides a means to generate diagnostic suites that thoroughly validate complex processors.

BVT TEST STRUCTURE

BVT provides a validation methodology that can be used from early pre-silicon development through post-silicon validation. To satisfy the needs of the development phase as well as compatibility testing of actual devices, several objectives are met.

Minimal Cycle Time

Functional tests are structured such that the amount of time needed to initialize, perform, and verify a function is as short as possible. Typical circuit simulators operate as slowly as two clock cycles per second, so the efficiency of the test programs is a critical factor in development productivity.

Self Checking Test Programs

The test program performs all checks for correct results and reports any errors. This removes the burden of manual checking of vectors or memory images, and enables automated regression testing.

Direct Failure Detection

The test programs indicate errors as directly as possible. A direct failure is one in which the test fails because the instruction or sequence of instructions being tested contains a failure. An example of a direct failure is "the ARPL test failed because the ARPL instruction incorrectly set condition codes." Direct failure detection focuses the designer on a specific problem in the design. An indirect failure occurs when a test fails not because the instruction or sequence under test fails, but because an instruction or sequence that the test uses for setup or checking results contains a failure. The test fails, but the reason for the failure can be obscured. An indirect failure might be "the processor took a General Protection exception prior to executing a task switch." Indirect failures can be much more difficult to identify than direct failures.

Consistency of Tests

Each test in the suite is structured consistently, regardless of the type of function under test. The means for specifying run-time test parameters and the method in which errors are reported should be consistent. The techniques the designer learns to debug the processor design should apply for all tests in the suite.

Implementation Independent Tests

The test programs do not presume any particular hardware implementation. They operate on models, simulations, emulations and devices. Each defined function is tested individually and validated in an implementation independent manner. Sequence tests are performed to validate dependencies between functions.

Repeatable Results

Each test behaves identically every time it is run, so that any errors can be analyzed. A random test that reports an error is of no use if the error cannot be replicated.

Isolation of Individual Test Cases

While a test program may exercise a large number of different test cases, it is structured such that any specific test case can be isolated and exercised individually. If a test program checks 100 test cases and only the ninety-ninth case fails, then the designer will want to focus on only the failing case in order to fix the problem. If the first 98 cases must always be exercised, then a great deal of time is spent testing something that is already working.

Bootstrapping Methodology

The validation suite is organized in a fashion such that the simplest functions are tested first. The testing of these functions does not require more complicated functions to be operational. Once a function has been verified, it can then be used to verify more complex functions. This "bootstrapping" structure allows the designer to implement and debug the design in an incremental fashion, rather than requiring him or her to implement complex microcode sequences prior to testing simple operations.

No Dependency on OS or System Resources

Test programs do not require the existence of operating systems or system resources so that they can be used very early in the development phase. In the early stages of development, simulations typically consist of only the processor and a simple memory, so the test programs must be able to operate in this minimal environment.

Simulation Environment Usage

In a pre-silicon simulation environment the designer has a simple means of loading test programs and initializing any run-time parameters. In the event of a failure, the test program reports the failure as quickly as possible. If the failing function and the detection of the failure are far apart, then the designer must spend a great deal of time backtracking in order to observe the failing event.

Post-Silicon Usage

When testing actual devices it can be assumed that most of the processor is operational and that system resources such as video monitors may be available. The test programs are able to be invoked in finished systems running operating systems, as well as in specialized prototype debug systems. The test program provides as much relevant information to the user as possible in the event of a failure. Because there is no longer direct visibility into the internals of the processor, the test program displays internal register contents, control register settings, and other pertinent information about the failing case.

BVT SUITE GENERATION

The goal of the BVT generated test suite is to test each processor function, from the simplest to the most complex, and verify its operation according to the instruction-set specification. There are many functions common to most processors, such as add, subtract, logical operations, and data movement instructions. For many of these functions, BVT provides tools that automatically generate operands and expected results which are embodied in the test programs. Layered upon these generic test cases are machine specific parameters such as register selection, memory addressing modes, and condition code rules.

Processor architectures diverge when it comes to more complex functions and operating modes. The most complicated instructions typically are unique to a given instruction-set. Often the behavior of a particular function will change depending on the operating mode of the processor. For instance, in the Pentium architecture, the far call instruction is a rather simple function when the processor is in real mode, but when in protected mode it becomes much more complicated. For example, the far call instruction may take one of several different actions, including exception cases, depending on the selectors and descriptors that it references. The development of test programs becomes very involved, and the test engineer can become bogged down in the details of coding these complex scenarios. As a result, the test programs for these complex functions can only be developed by the most experienced and knowledgeable programmers.

To address this problem, automatic code generation tools were developed as part of the BVT methodology. These code generation tools accept as input a description of various test scenarios. A description comprises the set of initial conditions, the function to be performed, and a set of expected results. The description is a high-level language that is separate from the low-level instruction coding. The code generation tools transform these descriptions into the actual machine code which will perform the tests and verify the results. This process frees the test writer from the drudgery of writing complex assembly language code and allows him or her to focus on the transformation of specification to test description.

BVT SUITE CONTENT

The basic organization of a behavioral validation suite for any processor is divided into three categories: basic function tests, corner case tests, and sequence tests. A validation suite must provide comprehensive tests in each of these areas.

Basic Function Tests

Basic function tests provide coverage of every processor instruction in all available modes. Basic tests focus on each instruction or function individually. These tests verify that the fundamentals of an instruction are working properly --- the proper result is calculated, registers and memory are updated correctly, and condition codes are set properly. Basic function tests also verify that unwanted side-effects do not exist.

Corner Case Tests

Corner case testing involves validating the boundary conditions and exception cases of a function. Many corner cases can be tested in an individual fashion, and this is done throughout the validation suite. Boundary conditions may involve certain operands that cause, for example, underflow or overflow in floating point operations, or that cause address calculations to cross memory boundaries. In addition, the processor architecture may define possible exceptions that an instruction can take. Each exception case that an instruction may allow is tested individually. This may include memory reference violations, page faults, privilege violations, and floating point exceptions. The x86 instruction-set architecture provides for a wide variety of exception cases, and for many functions multiple exceptions may exist concurrently. Of particular concern is the proper prioritization of multiple exceptions. For example, a task switch operation may contain a privilege violation for the code segment, an invalid new stack segment, and an instruction pointer that is beyond the code segment limit. The response to each of these error conditions is different, so the tests must verify that the proper action is taken.

Sequence Tests

Individual testing of functions ensures that each function operates properly in a stand-alone manner. Yet even if all functions work correctly individually, errors could occur when multiple functions are executed sequentially or concurrently. This is due to dependencies between instructions for register values, memory contents, and flag settings. In simple non-pipelined architectures, such dependencies are minimal or non-existent. But with more complex pipelined architectures such as the 486 and beyond, where many instructions may exist in various stages of execution at one time, additional circuitry is employed to increase performance. This circuitry introduces dependencies between instructions, which can lead to errors when executing sequences of instructions. Additional tests are necessary in the suite to verify that sequences of multiple instructions interact properly.

THE BVT586 VALIDATION SUITE

The BVT586 validation suite is the *Behavioral Verification Technology* based validation suite for Pentium class processors. It consists of 150 modules of Pentium assembly language instructions containing over 24,000 test programs, totaling over 900 megabytes of executable code. The suite checks the target processor for compatibility with the Pentium from the programmer's point of view. The test suite was calibrated for accuracy against the Intel Pentium programmer's reference manuals and the IEEE Floating Point specifications, other relevant manuals and actual Pentium devices. The suite's organization, interface and content reflect the feedback from hundreds of processor designers and validation engineers involved in 486 and Pentium compatible design projects.

BVT586 USAGE: PRE-SILICON

The BVT586 validation suite provides a specification-based tool for designing processors that are compatible to the Intel Pentium instruction-set architecture. The entire test suite can be run pre-silicon on any simulator with no requirements for additional x86 software (such as an operating system). When an error is detected by the suite, the test halts at the point of error and indicates to the user the failing test, failure condition and expected results. After design repair, the user can selectively rerun the failing test, sub-module(s), module(s) or the entire suite. The BVT586 suite can also run on hardware emulations of the device.

BVT586 USAGE: POST-SILICON

The same BVT586 suite is used post-silicon to test any device for differences in software perceived behavior from the Pentium processor. It is used by processor manufacturers and can be used by systems manufacturers. It is run as a DOS compatible program. Individual sub-modules can be run or sets of modules can be run as a regression test. Command line parameters allow control of error logging, error termination, test loop counts, and so forth. If a difference in operation between the device under test and the expected results is detected, the erroneous state and the expected state are reported. These identified differences are directly communicable between vendor and customer, validation engineer and designer, etc. This allows customers to participate intelligently in the validation and specification dialogue.

FUTURE VALIDATION CHALLENGES

The best situation for a processor or any complex electronic system is to specify the allowed behavior and hold the hardware and software accountable to that specification. BVT validation suites can be created to validate this specification for both software and hardware. This restriction of software to the specified system behavior will make the validation problem easier. Instead of all possible functions, only allowed behavior needs to be validated. This will also make system evolution easier since the new system has to maintain compatibility with only the specified behavior of the original. For life critical applications, this level of formal specification and formal validation is necessary now.

REFERENCES

Pentium™ Processor User's Manual, Vol. 3: Architecture and Programming Manual. 1994, Intel Corp.

Intel486™ DX Processor Programmer's Reference Manual. 1994, Intel Corp.

TESTING WITH APPLICATION SOFTWARE

To contrast against formal behavioral testing, we need to examine testing approaches that are not formally based on the specification. These approaches depend on running 'stress' applications and operating environments. Typically only a small percentage of the testing is done in simulation because simulation will not support operating systems and system applications. The majority of the testing is done in emulation and on silicon. This testing approach is very inefficient due to the limitations of system hardware emulation.

- High false error rate with hardware emulation since CPU must be run approximately 100 times slower than intended for the system.
- High false error rate caused by fragile mechanical assemblies involved in hardware emulation.
- High false error rate from the complexity of maintaining configurations of applications, boards, software patches and system limitations.
- High false error rate because commercial applications and operating systems are not test programs. They are complex, restricted in configurations and have their own bugs.
- Low test efficiency due to unmeasurable value of each test application.
- Tremendously long, inaccurate and complex debug process from perceived failure to identified cause. The debug effort involves debugging operating system and application from executable code only.
- Non-repeatable test results.
- Long reimplementation and retest time for repairs. This can be months for silicon and days or weeks if emulation. After reimplementation, all tests have to be rerun since it is impossible to know where side-effects may appear.
- Difficult to duplicate and maintain test environment.

Behavioral Verification Technology based formal verification of the instruction-set specification is orders of magnitude more effective in finding, identifying, repairing and retesting bugs than one based on verification of system level applications.

RePent™ – *Peace of mind for Pentium users*

Matt Trask
Communica, Inc.
118 Waterhouse Rd., Ste. B
Bourne, MA 02532
Ph. (508) 759-6714 Fax (508) 759-7812
E-Mail: matt.trask@bix.com

Matt Trask is President of Communica, Inc., a system software firm based on Cape Cod that provides OEM development services to computer system vendors. His background includes various virtual machine and protect-mode operating system, development project on the Intel CPU Family. Communica's RePent™ software detects and corrects the Pentium FPU defect, preventing damage to floating point data.

Communica

RePent™

Peace of mind for Pentium users

Are you worried that the defect in your Pentium CPU could damage your data?

Are you concerned that you can't detect defective floating point calculations?

Are you unwilling to wait perhaps months before Intel can replace your defective Pentium chip?

Communica has the simple solution to your worries: RePent™. This is a special program that runs on your Pentium computer and monitors all use of the Floating Point Unit (FPU). Any attempt to use a defective instruction is trapped by the software and calculated by RePent without using the defective part of the FPU. More importantly, RePent notifies you that this has occurred and that you have been protected from inadvertant damage to your data. RePent has been carefully designed to minimize overhead while providing complete protection from defects in your Pentium FPU.

RePent for Microsoft Windows will be shipping in mid-January, 1995 followed by versions for IBM's OS/2 and Microsoft Windows NT. Single unit pricing is $39 for each of these operating systems with quantity discounts and site licenses available. Support for other operating environments is available on special request. For more information, or to place an order for Repent, call Communica at 800/2-FIX-BUG or 800/FIX-A-586.

[02/06/95]

RePent^{TM}

A software fix for Pentium FPU defects

Matt Trask, Communica, Inc. 13 Dec 94

Overview

Current versions of the Intel Pentium CPU are known to contain significant defects in their Floating Point Units (FPUs) that can affect common math operations such as division, remainder, and some transcendental math operations that use the division hardware. The nature of the problem is such that it is unlikely to occur in most uses of a Pentium system, but the problem is entirely deterministic. In other words, if it happens once during a floating point calculation, it will occur every time the same calculation is performed.

Communica has developed a software solution to this problem that is designed to minimize performance overhead while providing complete protection from accidental data corruption by the FPU defect. This document describes Communica's software solution and characterizes system performance while the software is in use.

The Problem

The Pentium uses a new algorithm for performing floating point divides that is based on lookups in a large table. According to Intel's document[1] that describes the Pentium bug, "the cause of the problem traces itself to a few missing entries in a lookup table used in the hardware implementation algorithm for the divide operation. Since this divide operation is used by the Divide, Remaindering, and certain Transcendental Instructions, an inaccuracy introduced in the operation manifests itself as an inaccuracy in the results generated by these instructions."

Other Solutions

The first and most obvious solution was the approach taken by Compaq Computer Corp: the Pentium FPU can be disabled under software control, preventing inadvertent use of defective floating point operations by users. This solution is simple and safe, but highly undesirable because it causes a significant performance penalty to users of floating point operations.

Lotus and Microsoft have taken an equivalent approach, giving users instructions that explain how to configure their products to ignore the FPU and use their built-in floating point calculation routines. From a performance perspective, this is no different than what Compaq has done.

Intel has formed a working group to study this problem and has issued a software workaround that provides for efficient execution of floating point division that does not use the FPU. Unfortunately, Intel's method of supplying this solution to Pentium users has been to give it away to compiler writers for inclusion in run-time libraries as a replacement for routines that currently use the FPU. This is a technically elegant solution, but it requires software vendors to rebuild their applications and provide updated versions to their customers, thus incurring potentially significant expense for retesting and redistribution. This method is not a general purpose solution - even when some of a user's programs have been updated, other applications may still be at risk from the defect.

As of 20 Dec 94, Intel has publicly committed to exchange any defective CPU at the owners request for the life of the system. While this is the best possible solution, the logistics of this process will probably take many months or even years, and even then, there is no assurance that all Pentium CPUs will be replaced. A future owner or user of a Pentium system may still have a defective CPU and have no way of knowing this.

Communica's Solution

Our approach to solving this problem is focused on ease of use and efficiency. Each floating point operation is trapped and examined - if it is one of the defective instructions, the calculation is performed in software. If the opcode is not on the list of known bad instructions, the trap is released as quickly as possible, permitting execution on the FPU hardware.

The most important feature of Communica's design is the notification component. If a user's applications actually access the Pentium FPU, the software can be configured to notify the user immediately, thus removing all reasons for concern from unsophisticated users who are not at risk of damage to their data. This software is an excellent means for a system vendor to qualify and prioritize Pentium owners for CPU replacement based on demonstrable need.

At this time, Communica's software runs as a TSR extension to DOS. Implementations for Windows and OS/2 are currently under construction. Current plans call for development of versions for Windows NT, Unix and Windows 95. If there is sufficient interest, we will also develop a Netware NLM version.

Performance Considerations

The current algorithm used in Communica's software adds a fixed overhead of approximately 37 assembler-level instructions to each floating point instruction before it is released for execution on the FPU. There is no CPU overhead at all for users that do not use software that accesses the FPU. Because of this overhead, Communica's software categorizes users into three groups: non-FPU users that have no performance overhead and who need not worry about damage to their data because they can be notified if they ever use a new program that uses the FPU, light users of the FPU who don't use it often enough to worry about the additional overhead and still need not worry about damage to their data, and heavy FPU users who probably would be concerned about the performance overhead, but can use the evidence provided by running Communica's software to justify expeditious replacement of their Pentium chip.

We recognize that the performance overhead caused by this solution is undesirable and are working to reduce this as much as possible. At this time we have a new algorithm that we believe can lower the overhead to as little as five or six assembler-level instructions. Development of a proof-of-concept is currently under way.

Company Background

Formed in 1989, Communica is a team of senior system programmers and electrical engineers that provides OEM development services to most of the major personal computer vendors. Areas of expertise include low-level system software such as PC-compatible BIOS, device drivers, and communication and networking software. Virtual Machine Technology and device emulation on Intel-family CPUs are a specialty area of expertise. Communica's engineering staff has considerable experience working with DOS/Windows internals, Unix kernel programming, and low-level system programming for OS/2, Windows NT, and Netware. Communica's client list includes IBM, NCR/AT&T, Sun Microsystems, Stac Electronics, Central Point Software, and many others.

In addition to providing development services, Communica has licensed the IBM SurePath™ BIOS as the basis for developing upgrade products that enhance existing computer systems by adding support for Plug'n'Play and ReZoom™. Communica's ReZoom is a technology for enabling desktop systems to suspend operation and rapidly resume at a later time similar to the way most laptop computers operate. ReGreen™ is Communica's external power controller that is used to add EPA EnergyStar functionality to existing computer systems.

For more information on RePent™, Communica's software solution to the Pentium defect, or other products or services provided by Communica, call 508/759-6714 or send email to *matt.trask@bix.com*. Pentium users that wish to order RePent for delivery in mid-January, 1995, please call 800/FIX-A-586 or 800/2-FIX-BUG after 27 Dec 94. Single quantity pricing is $39 with reseller and quantity discounts, and OEM and site licensing available.

[1] Statistical Analysis of Floating Point Flaw in the Pentium™ Processor (1994), Intel Corporation, 30 Nov 94, by H.P. Sharangpani and M.L. Barton

Thermal Issues in Working with Pentium (Registered Trademark)

Gary Kuzmin
Director of Corporate Technical Marketing
Aavid Thermal Technologies Inc.
One Kool Path
Laconia, NH 03247
(603) 528-3400

As large-scale integration has expanded, thermal managment has become a critical issue for processor and system vendors alike. New thermal solutions are required, because the performance and reliability of the semiconductor is constrained by temperature. The daunting task is to figure the best solution with the assurance that the customer is going to have a reliable system.

PCI Graphics Boards

Jack Roberts
Director and Principal Analyst
Graphics and Displays
Dataquest Incorporated
1290 Ridder Park Drive
San Jose, CA 95131
Phone: (408) 437-8539
Fax: (408) 437-0292
E-mail: jroberts@dataquest.com

The adoption of the PCI bus across multiple desktop computer platforms, i.e. Intel-based PCs, Macintoshes, and technical workstations, will have a dramatic effect upon the graphics boards market. No longer will workstation users have high-end graphics performance as their exclusive domain. No longer will the Macintosh user be forced to pay a premium for graphics boards compared to users of PCs. This leveling of the graphics playing field promises to create a highly competitive market where only the strong or tightly niched are able to survive. This session looks at the opportunities and obsticales facing the graphics boards market.

A 192-Bit Graphics Controller for the PCI Marketplace

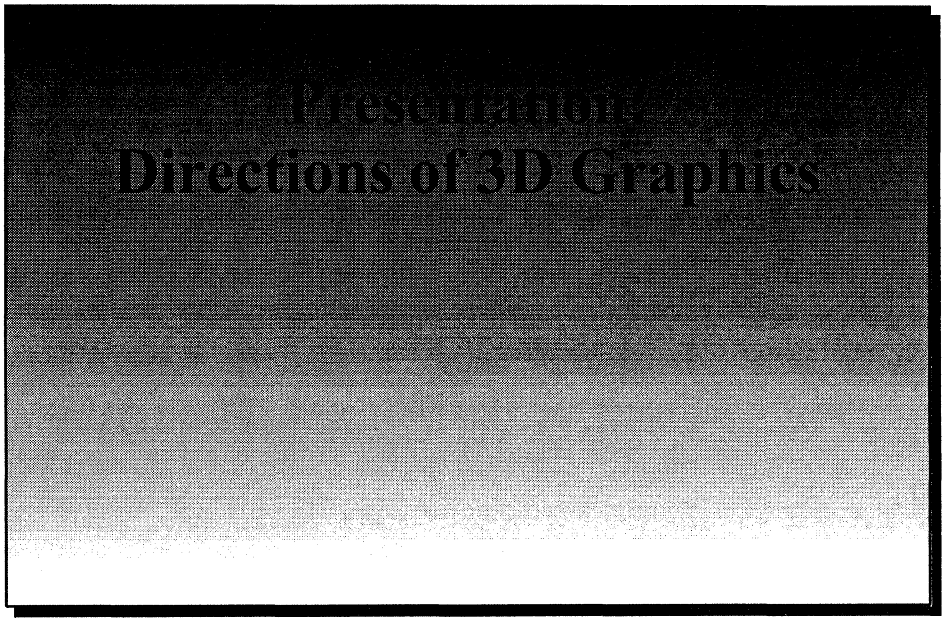
Joe Eschbach
General Manager
rPC
215 Moffett Park Drive
Sunnyvale, CA 94089
(408) 541-5400/5672 (fax)

rPC has developed the FireStorm192 with a unique architectural design that offers the user a 192-bit wide data path, providing true color (16.7 million colors) at resolutions up to 1600 X 1200. FireStorm192's features are ideal for creative professionals who need a high level of graphics performance when using desktop publishing applications such as Adobe PhotoShop, Quark Xpress, and Aldus PageMaker.

rPC is an independent business unit of Radius Inc., focused on the Windows-based graphics market.

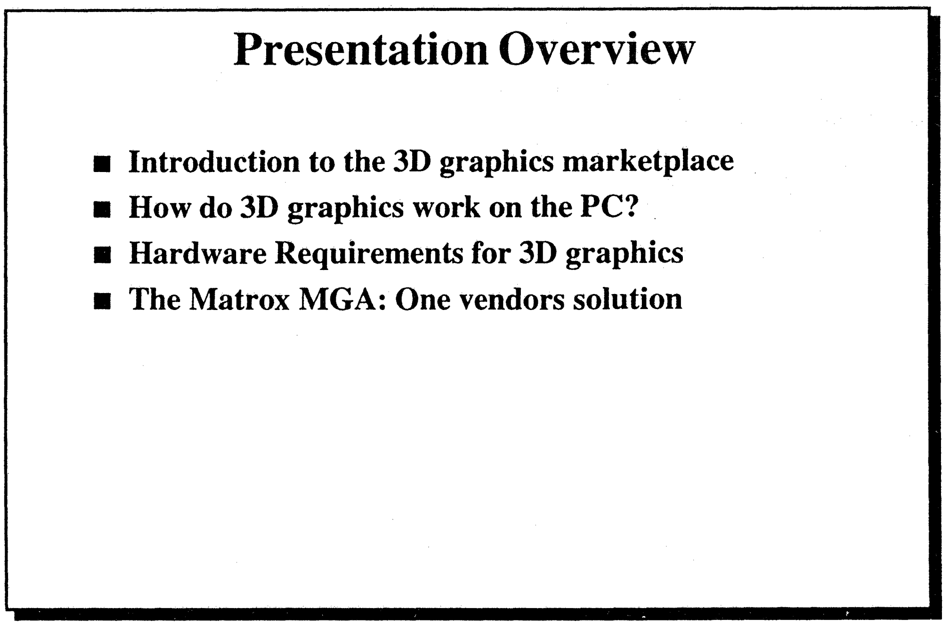
MATROX MGA™ POWER GRAPHICS

Lorne Trottier
President
Matrox Graphics Inc.
1055 St. Regis Blvd.
Dorval, H9P 2T4, Quebec, CANADA
Ph. (514) 685-2630 Fax (514) 685-2850



Presentation
Directions of 3D Graphics

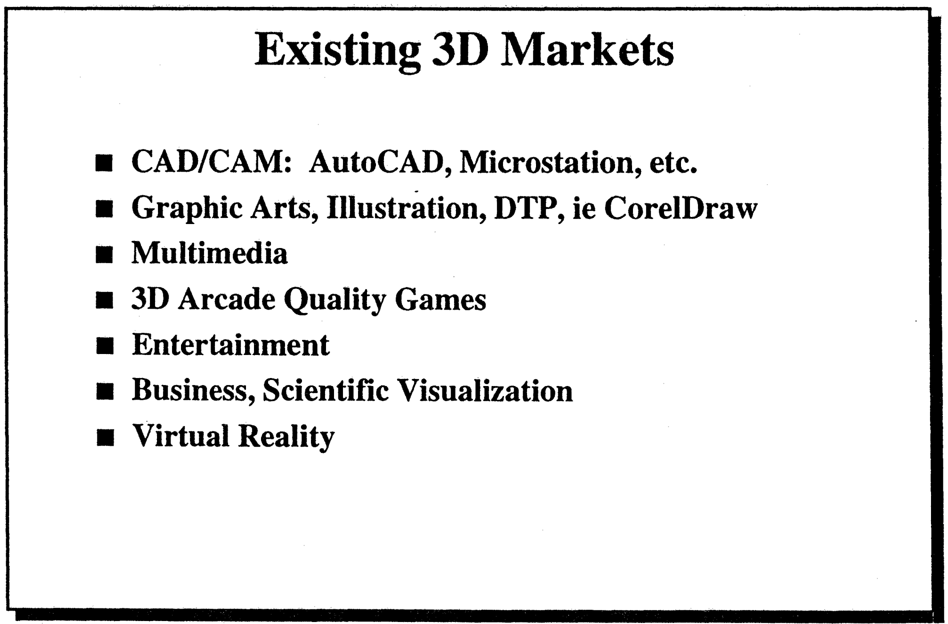
PCI Conference



Presentation Overview

- Introduction to the 3D graphics marketplace
- How do 3D graphics work on the PC?
- Hardware Requirements for 3D graphics
- The Matrox MGA: One vendors solution

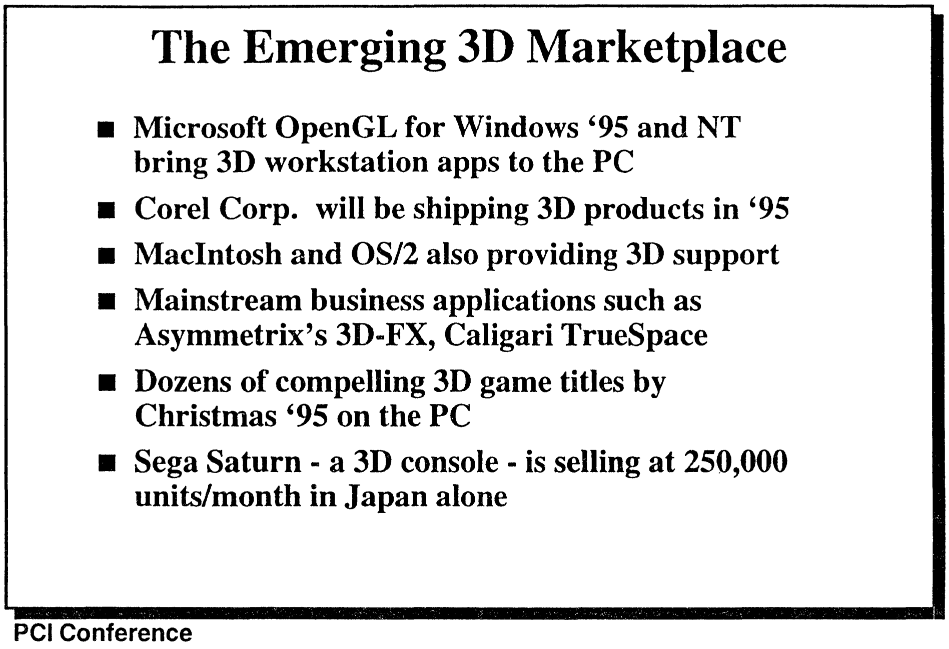
PCI Conference



Existing 3D Markets

- CAD/CAM: AutoCAD, Microstation, etc.
- Graphic Arts, Illustration, DTP, ie CorelDraw
- Multimedia
- 3D Arcade Quality Games
- Entertainment
- Business, Scientific Visualization
- Virtual Reality

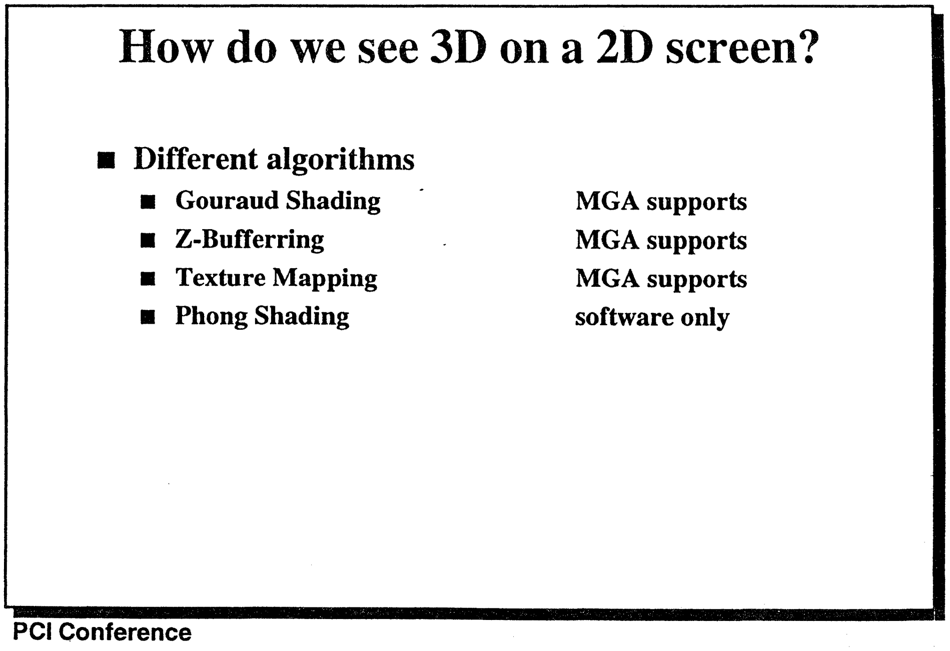
PCI Conference



The Emerging 3D Marketplace

- Microsoft OpenGL for Windows '95 and NT bring 3D workstation apps to the PC
- Corel Corp. will be shipping 3D products in '95
- MacIntosh and OS/2 also providing 3D support
- Mainstream business applications such as Asymmetrix's 3D-FX, Caligari TrueSpace
- Dozens of compelling 3D game titles by Christmas '95 on the PC
- Sega Saturn - a 3D console - is selling at 250,000 units/month in Japan alone

PCI Conference



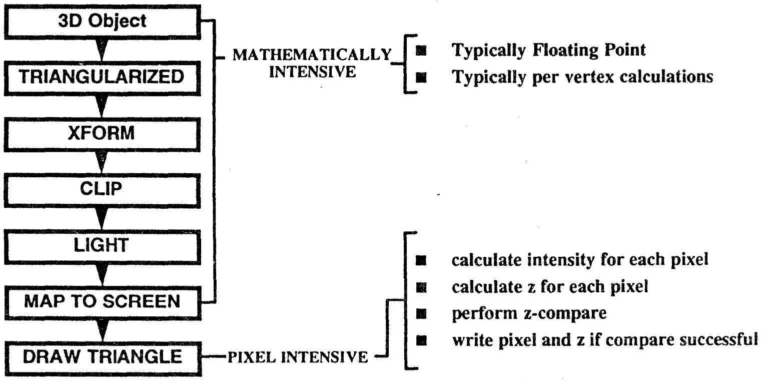
How do we see 3D on a 2D screen?

- Different algorithms

Gouraud Shading	MGA supports
Z-Bufferring	MGA supports
Texture Mapping	MGA supports
Phong Shading	software only

PCI Conference

Typical 3D Pipeline for Shading



3D Object

TRIANGULARIZED

XFORM

CLIP

LIGHT

MAP TO SCREEN

DRAW TRIANGLE

MATHEMATICALLY INTENSIVE

- Typically Floating Point
- Typically per vertex calculations

PIXEL INTENSIVE

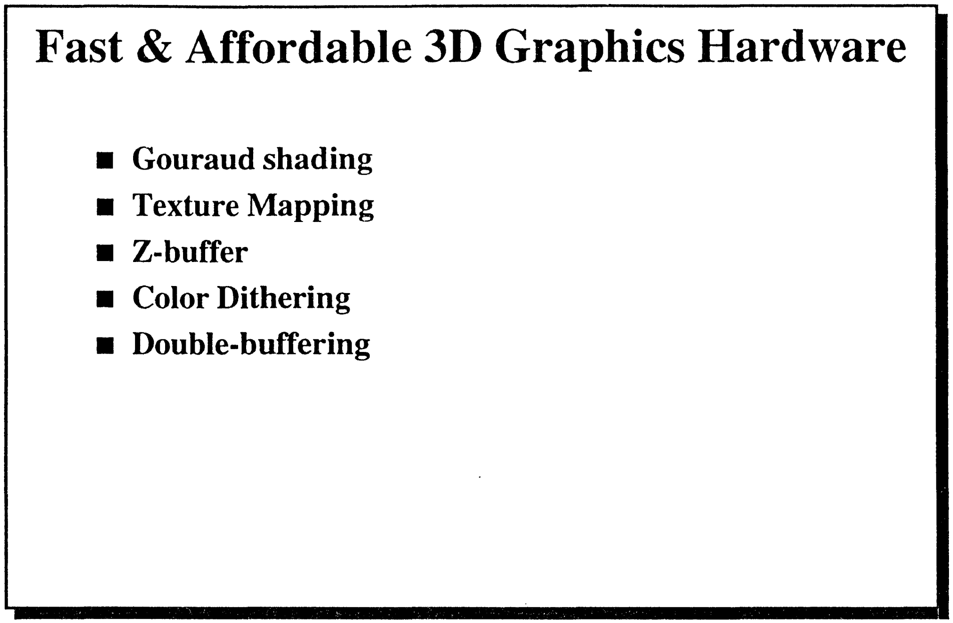
- calculate intensity for each pixel
- calculate z for each pixel
- perform z-compare
- write pixel and z if compare successful

PCI Conference

Realtime 3D Required Elements

- High Performance Floating Point Systems
- Fast and affordable 3D graphics controller
- Fast frame buffer memory type
- System bus with high data rate
- Software standards for 3D

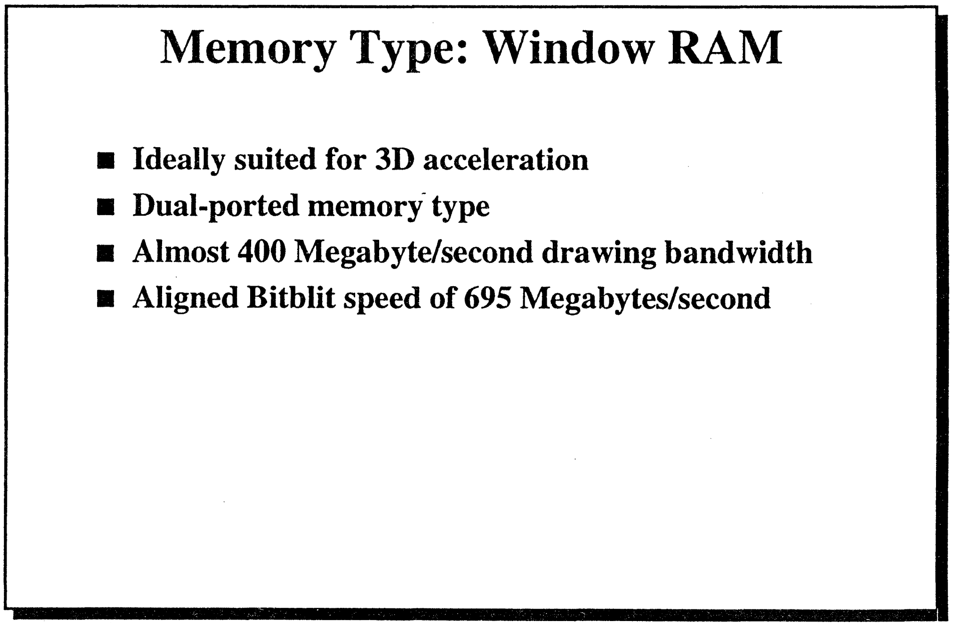
PCI Conference



Fast & Affordable 3D Graphics Hardware

- Gouraud shading
- Texture Mapping
- Z-buffer
- Color Dithering
- Double-buffering

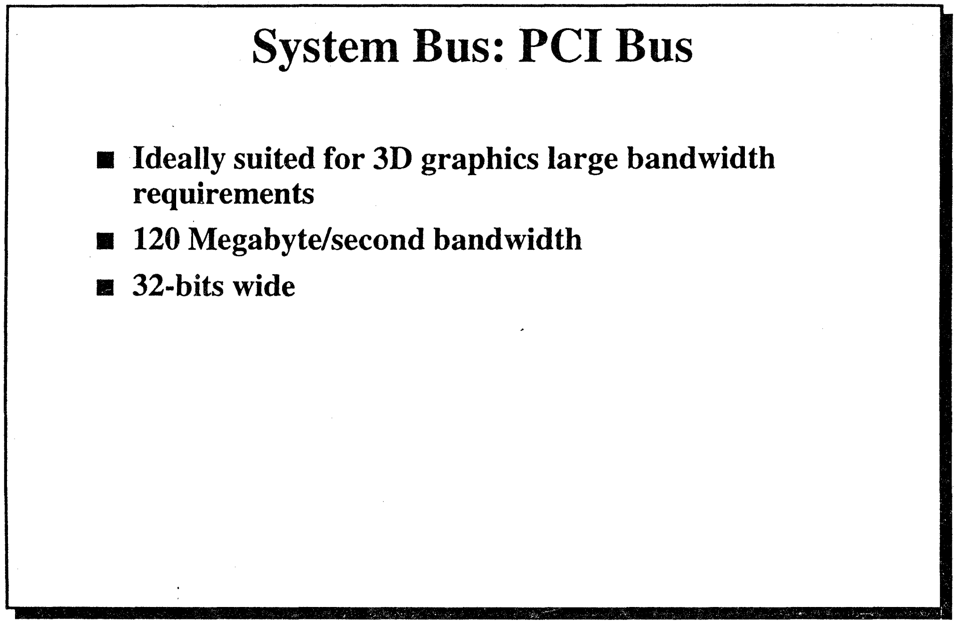
PCI Conference



Memory Type: Window RAM

- Ideally suited for 3D acceleration
- Dual-ported memory type
- Almost 400 Megabyte/second drawing bandwidth
- Aligned Bitblit speed of 695 Megabytes/second

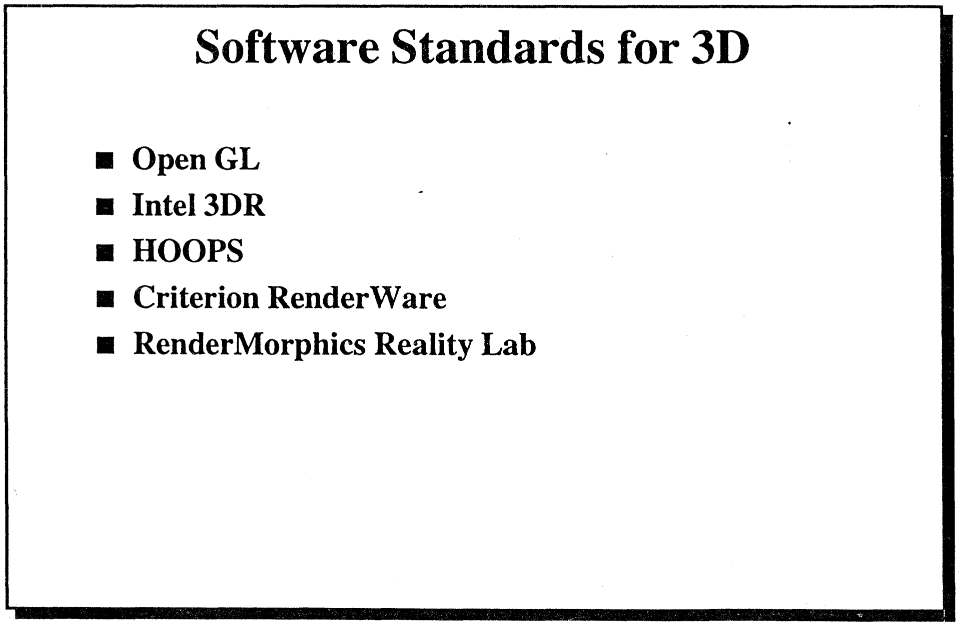
PCI Conference



System Bus: PCI Bus

- Ideally suited for 3D graphics large bandwidth requirements
- 120 Megabyte/second bandwidth
- 32-bits wide

PCI Conference

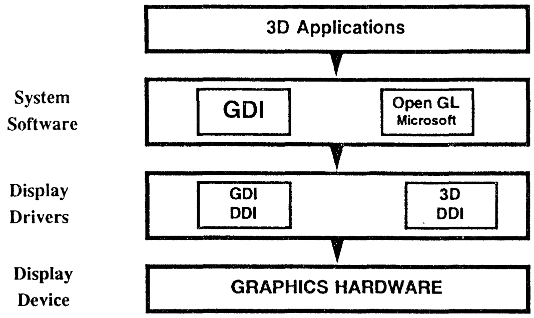


Software Standards for 3D

- Open GL
- Intel 3DR
- HOOPS
- Criterion RenderWare
- RenderMorphics Reality Lab

PCI Conference

Open GL & Microsoft NT



3D Applications

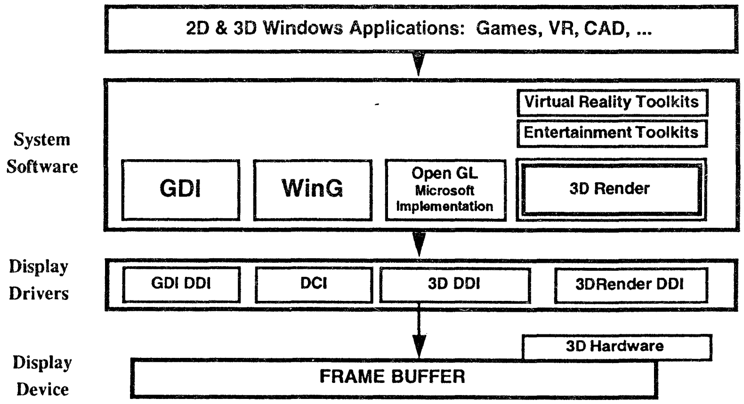
System Software: GDI | Open GL Microsoft

Display Drivers: GDI DDI | 3D DDI

Display Device: GRAPHICS HARDWARE

PCI Conference

Intel 3DR



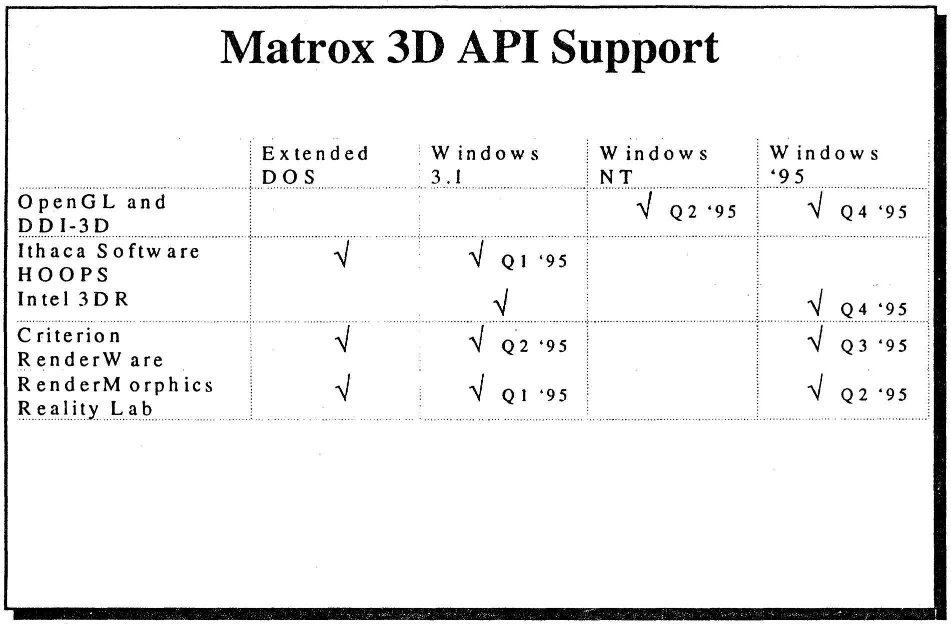
2D & 3D Windows Applications: Games, VR, CAD, ...

System Software: Virtual Reality Toolkits | Entertainment Toolkits | GDI | WinG | Open GL Microsoft Implementation | 3D Render

Display Drivers: GDI DDI | DCI | 3D DDI | 3DRender DDI

Display Device: 3D Hardware | FRAME BUFFER

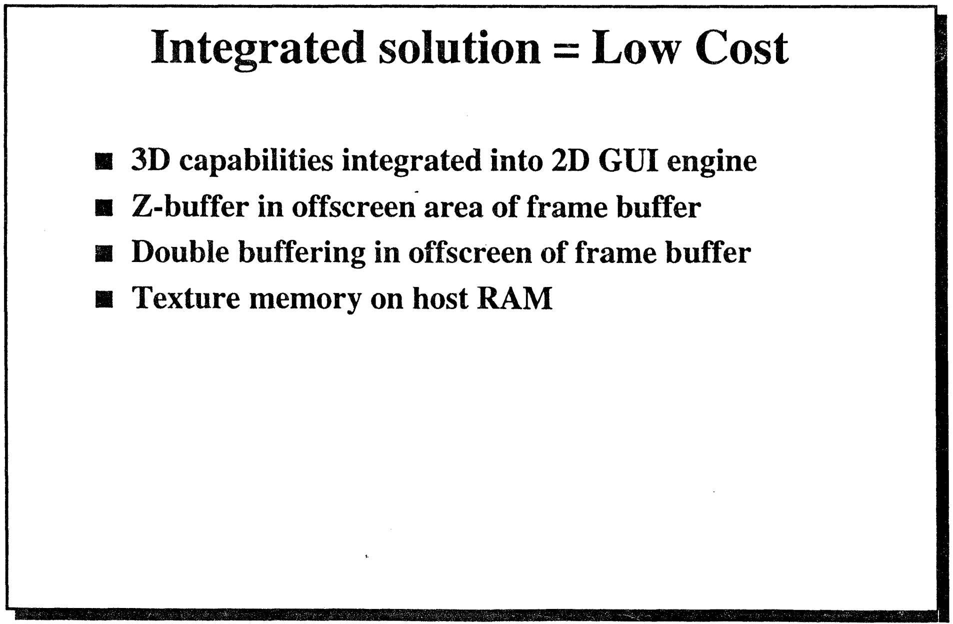
PCI Conference



Matrox 3D API Support

	Extended DOS	Windows 3.1	Windows NT	Windows '95
OpenGL and DDI-3D			√ Q2 '95	√ Q4 '95
Ithaca Software HOOPS	√	√ Q1 '95		
Intel 3DR		√		√ Q4 '95
Criterion RenderWare	√	√ Q2 '95		√ Q3 '95
RenderMorphics Reality Lab	√	√ Q1 '95		√ Q2 '95

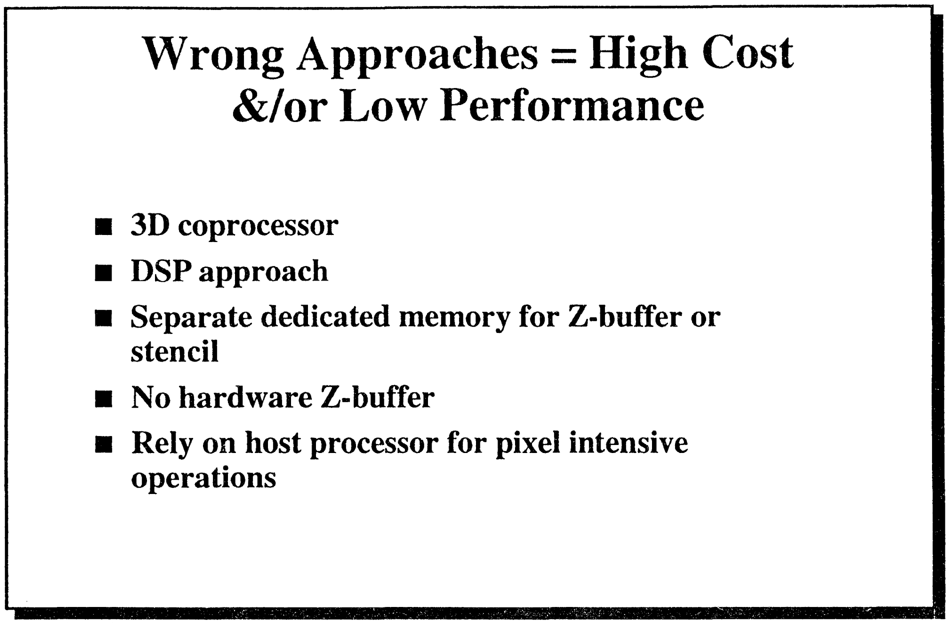
PCI Conference



Integrated solution = Low Cost

- 3D capabilities integrated into 2D GUI engine
- Z-buffer in offscreen area of frame buffer
- Double buffering in offscreen of frame buffer
- Texture memory on host RAM

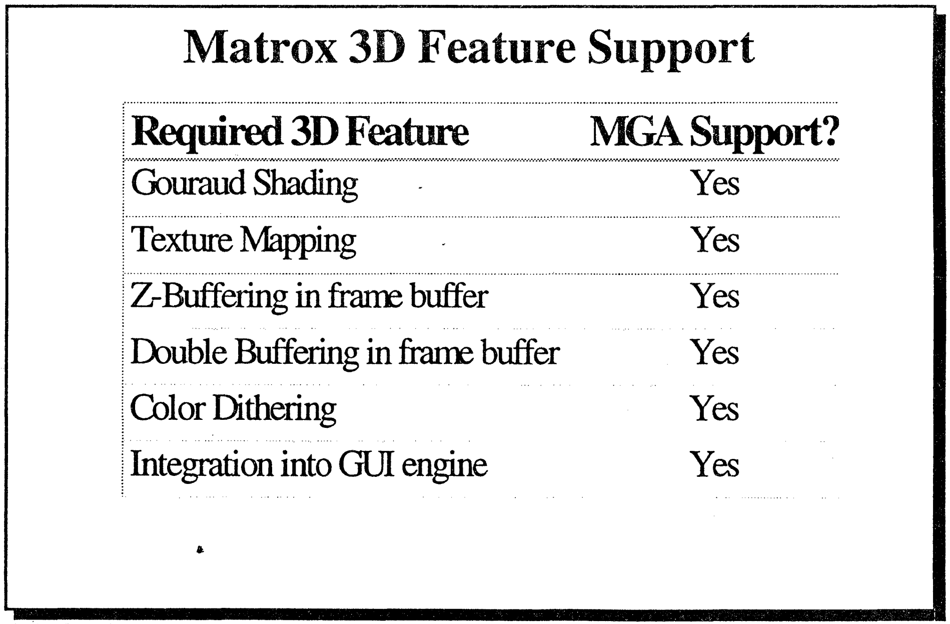
PCI Conference



Wrong Approaches = High Cost &/or Low Performance

- 3D coprocessor
- DSP approach
- Separate dedicated memory for Z-buffer or stencil
- No hardware Z-buffer
- Rely on host processor for pixel intensive operations

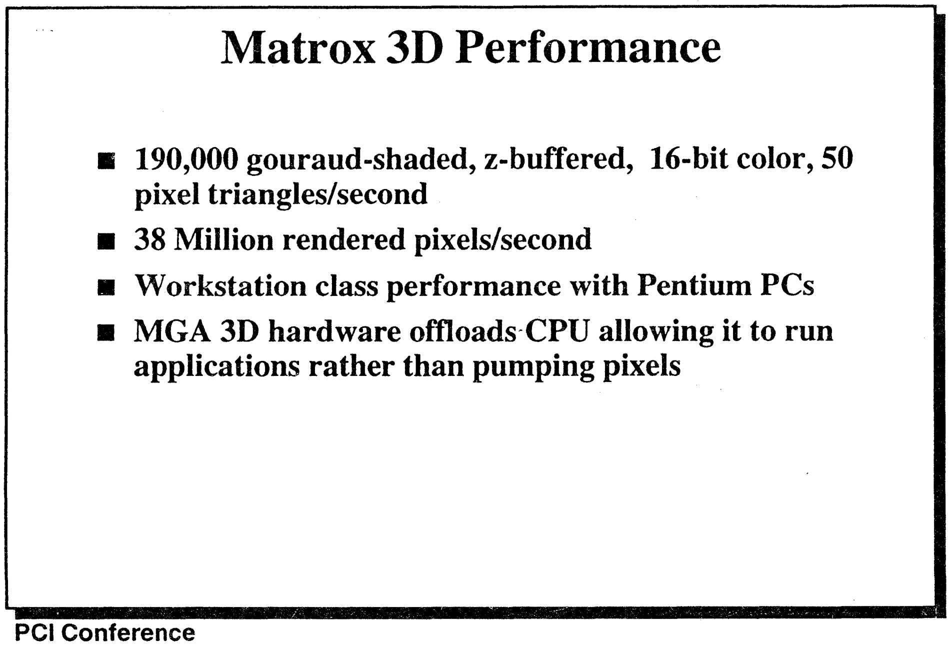
PCI Conference



Matrox 3D Feature Support

Required 3D Feature	MGA Support?
Gouraud Shading	Yes
Texture Mapping	Yes
Z-Buffering in frame buffer	Yes
Double Buffering in frame buffer	Yes
Color Dithering	Yes
Integration into GUI engine	Yes

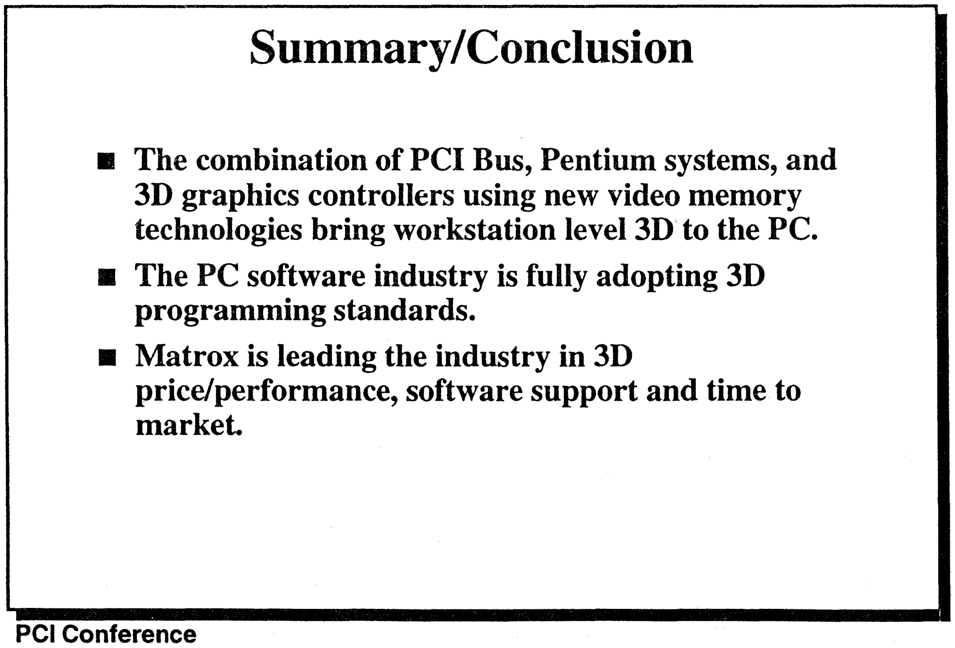
PCI Conference



Matrox 3D Performance

- 190,000 gouraud-shaded, z-buffered, 16-bit color, 50 pixel triangles/second
- 38 Million rendered pixels/second
- Workstation class performance with Pentium PCs
- MGA 3D hardware offloads CPU allowing it to run applications rather than pumping pixels

PCI Conference



Summary/Conclusion

- The combination of PCI Bus, Pentium systems, and 3D graphics controllers using new video memory technologies bring workstation level 3D to the PC.
- The PC software industry is fully adopting 3D programming standards.
- Matrox is leading the industry in 3D price/performance, software support and time to market.

PCI Conference

Graphics, Motion Video, and PCI

Michael Hawkey
Marketing Manager

Multimedia Products Unit
Western Digital Corporation
800 E. Middlefield Road
Mountain View, CA 94043

hawkey_m@a1.wdc.com
Phone: 415-335-2590
Fax: 415-335-2515

Abstract

Western Digital is approaching the display controller add-in board market from the point of view of the PCI bus. "What data types does the PCI bus pass to the display controller?" "Is the system only driving data on the PCI bus to the display controller or is the display controller talking back?" These questions, and the interaction of the PCI bus with new functions like 3D and Motion Video acceleration, are explored at both the technical and marketing levels.

GRAPHICS CARD DESIGN: SYSTEM IMPLICATIONS AND CONSTRAINTS

Billy Garrett
Manager of Graphics Development
Rambus Inc.
2465 Latham Street
Mountain View, CA 94040
garrett@rambus.com

ABSTRACT

Since the introduction of Windows™ 3.1 with its standardized application interface, graphics cards for personal computers have advanced significantly in price/performance and feature sets, as they take advantage of new silicon technology. While the PC market moved to 1024 x 768 displays that use at least 1MByte frame buffers, DRAM frame buffer cost has exceeded controller cost in most card designs. As frame buffers on graphics cards continue to expand, the memory component costs dominate the cost of the graphics subsystem. The system requirements of PCI (or VL) bus interfaces, directly integrated RAMDAC, synthesizer technology, and additional graphics or video buses (feature connector, VAFC, and VESA Media Channel) put a significant pin count burden on the controller. The challenge for the graphics card designer is to implement the frame buffer with the most cost-effective memory technology while supporting the required bandwidth and feature sets for high quality displays and multimedia applications. Rambus™ 500MHz, low pin-count memory technology meets these system requirements.

THE IMPACT OF WINDOWS ON GRAPHICS CARDS

As PC software moved from character-based to Windows applications, the amount of data that needed to be manipulated on the screen increased dramatically. In the original IBM CGA graphics card, only 2KBytes of data was required to specify an entire screen of information, and that format was widely used by applications of the day: Lotus 123, Wordstar, DBase, and many others. Windows and Windows applications have changed all that; applications like Word, Excel, and Powerpoint now make use of the pixel-mapped surface of the display. At a minimum, these applications require VGA; at a resolution of 640 x 480 x 8bpp, over 300KBytes of information must be written to the frame buffer to completely update the screen. Today, most PCs ship with frame buffers that are 1MByte or larger in order to support 800 x 600bpp or 1024 x 768bpp display resolutions. These resolutions require somewhere between 512KBytes and 768KBytes of data for the display. This represents a 256 to 384 fold increase in display buffer size, in the last 12 years.

The move to Windows applications has had several effects on the graphics subsystem. Most PCs are now sold with some type of Windows GUI (Graphics User Interface) accelerator chip. Graphics controller companies such as Cirrus Logic, S3, ATI, Matrox, Tseng Labs, and Trident have added functionality to their VGA cores to accelerate Windows operations, usually in the form of a bit-blit data path or controller, and sometimes font/color expansion and line drawing engines. This new VGA functionality plus a first-rate Windows driver significantly increases performance over the original VGA chips.

The increase in frame buffer size has reached the point where the frame buffer memory cost exceeds the cost of the GUI controller. Now that memory is the dominant cost item of the graphics board, the designer looks for ways to minimize the frame buffer cost while meeting required performance. The DRAM alternatives for graphics frame buffers are discussed in a later section.

As more multimedia applications become available for Windows PCs, GUI controllers will gain more features, such as video overlay capability, 3D support, additional acceleration features for Windows, MPEG decompression, and audio support. Each of these functions further increases the frame buffer size and required bandwidth.

SYSTEM ISSUES: COST FACTORS

A designer must consider many system issues to deliver a successful graphics card: competitive cost, performance, features, reliability, time-to-market, and so on. Some of these issues are beyond the scope of this paper; however, because card costs and performance contribute the most toward the success of the end product, they will be examined here. The designer needs to evaluate several factors such as component costs (particularly the frame buffer DRAMs and graphics controller chip), memory expansion support, board space and EMI. Secondary factors are costs for assembly, testing, inventory, handling, and so on. As mentioned before, display resolution support determines the frame buffer size and cost, and must be considered first.

SYSTEM ISSUES: DISPLAY RESOLUTION SUPPORT

Mainstream PCs have moved to support 1024 x 768 displays, and PC users want the ability to display 256 or more colors. Most PCs are sold today with 1MByte frame buffers and can accept add-in graphics cards supporting at least 2MByte frame buffers. To accommodate these directions, in the design phase graphics designers try to create a frame buffer memory architecture that can support one to several megabytes.

The card's maximum advertised display resolution and the number of bits per pixel (bpp) determine the amount of memory required for the frame buffer, the required frame buffer bandwidth, and the speed of the RAMDAC. If any of these three features is deficient, the maximum display resolution, refresh rate, or pixel depth cannot be supported. Table 1 summarizes these relationships.

Table 1: Display Resolution versus Memory Required, RAMDAC Speed and Bandwidth

Display Resolution	KB of Memory	DAC Speed	Max Bandwidth
640 x 480 x 8	300	31.5 MHz	31.5 MB/s
640 x 480 x 16	600		63 MB/s
640 x 480 x 24	900		94.5 MB/s
800 x 600 x 8	469	50 MHz	50 MB/s
800 x 600 x 16	938		100 MB/s
800 x 600 x 24	1,407		150 MB/s
1024 x 768 x 8	768	80 MHz	80 MB/s
1024 x 768 x 16	1,536		160 MB/s
1024 x 768 x 24	2,304		240 MB/s
1280 x 1024 x 8	1,280	135 MHz	135 MB/s
1280 x 1024 x 16	2,560		270 MB/s
1280 x 1024 x 24	3,840		405 MB/s
1600 x 1200 x 8	1,875	170 MHz	170 MB/s
1600 x 1200 x 16	3,750		340 MB/s
1600 x 1200 x 24	5,625		510 MB/s

Table 1: Display Resolution versus Memory Required, RAMDAC Speed and Bandwidth

In order to achieve the bandwidth necessary to support these display resolutions using conventional DRAMs, designers have used two or four DRAM components in 32- or 64-bit wide data buses. These conventional DRAMs present a granularity issue. For example, implementing a 64-bit bus requires four x16 DRAMs (DRAMs with 16-bit I/O). Using 4Mbit DRAM technology, this leads to using four 256Kx16 DRAMs (page mode, EDO or SDRAM) adding up to a 2MByte frame buffer. A 1MByte frame buffer uses two DRAMs in a 32-bit bus. Most 64-bit graphics controllers use only a 32-bit bus when configured with 1MByte of memory. Most consumers are unaware that a 2MByte frame buffer is required to take full advantage of the card's advertised bandwidth and rated performance.

Graphics Frame Buffer Cost Factors

Since the frame buffer implementation dominates overall component costs, the choice of DRAM device is critical. This requires some homework as there are several DRAM types available for graphics subsystems. DRAM prices vary from vendor to vendor and with the organization and speed grade of the memory device. This year 16Mbit DRAM technology becomes lower cost on a per-bit basis than previous generation of 4Mbit DRAMs, so that the most cost effective frame buffers will be implemented with 16Mbit technology. In addition, the DRAM choice impacts controller cost, since the connection to the DRAM array determines the pin count and, potentially, the die area of the controller.

GRAPHICS DRAM ALTERNATIVES

Until recently, there were only two DRAM choices: page mode DRAMs or video-RAMs (VRAM). Today, the number choices has greatly increased. Page mode DRAMs are being replaced with Extended Data Out (EDO) DRAMs, which provide added bandwidth by reducing page mode cycle times. Synchronous DRAMs (SDRAMs) and Synchronous Graphics RAMs (SGRAMs) attempt to solve the bandwidth issue by adding a new synchronous interface to a standard DRAM core. Denser VRAMs, Window-RAMs (WRAMs), and Synchronous

VRAMs (SVRAMs) are available for dual-ported frame buffers. The Rambus DRAM (RDRAM™) represents a revolutionary approach to increasing bandwidth. RDRAMs transfer data at a 500MHz rate over a narrow, byte-wide bus, referred to as the Rambus Channel.

Extended Data Out (EDO)

EDO DRAMs are like conventional page mode DRAMs, except that the way in which data is disabled on a read is changed from the rising edge of CAS to WE. The outputs are held when CAS rises. This combined with a few other changes allow EDO DRAMs to cycle faster in page mode, therefore offering additional bandwidth. The signals RAS, CAS, WE, and OE remain the same as for a page mode DRAM, making the design transition to EDO straightforward. In order to provide sufficient bandwidth for graphics applications, the wide 16-bit I/O versions of these DRAMs are available in the 4Mbit generation. A 16Mbit page mode or EDO DRAM is not suitable for graphics due to granularity issues. Although a 16Mbit DRAM provides sufficient storage for a 2MByte frame buffer, it does not provide enough performance to meet associated display requirements.

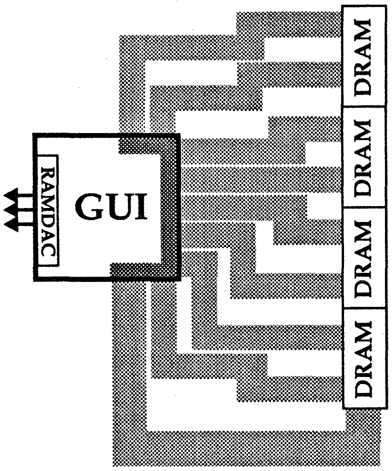
SDRAM and SGRAM

SDRAMs have an evolutionary design compared to conventional DRAMs. Internally, they are arranged in two banks, each independent and holding half of the DRAM bits. Available in a variety of bus widths (x4, x8, and x16), these devices are applicable for graphics at the 4Mbit (x16) and 8Mbit SGRAM (x32) densities. Although 16Mbit SDRAMs are becoming more available this year, the 16-bit wide bus is not fast enough to support the display refresh performance requirements of the majority of important display sizes.

While the interface to an SDRAM appears similar to a conventional DRAM, the timing and "commands" sent to a SDRAM are different from the RAS/CAS timing normally associated with a DRAM. The graphics controller designer must develop a new state-machine in order to support SDRAMs.

SGRAMs, which are based on SDRAMs, are offered in the 8Mbit density. They use a x32 interface in order to provide higher bandwidth for graphics. SGRAMs include the block write function, which allows SGRAMs to write as much as 32 bytes in parallel, but only every other clock cycle. They also have a single-color register, so that color expansion requires two passes for a font. Block write adds an increase in bandwidth of up to four times for pattern fills and up to two times for fonts. In addition to the wider bus, SGRAMs have one more pin, DSF, that is used to encode "commands" to the SGRAM.

Most SDRAM vendors offer parts that operate up to 66MHz. Data sheets have become available for SGRAMs, with some vendors showing 100MHz bin split parts. Since parts are not yet in production, yield on these parts has not been established. Achieving 100MHz operation on a board, using LVTTL signaling will be very difficult, because of board layout, clock/data trace, and skew issues. Memory expansion sockets further complicate the board design such that SGRAM configurations supporting expandable frame buffers are not expected to achieve the component's specified 100MHz operation in a system environment.



Single-Ported DRAM Approach

VRAM and WRAM

VRAMs are available in up to 4Mbit densities. The 4Mbit parts are arranged as x16 devices (for both parallel and serial ports) and are contained in 64-pin packages. Because of the their dual-port design and other features, VRAMs have cost up to twice that of single-ported DRAMs, and, thus, have been used only in high performance add-in cards. Most current card designs are moving away from VRAMs because of cost and the additional pin count incurred due to the second port.

Samsung offers the WRAM, a special version of an 8Mbit VRAM, which is intended to be priced 40%-50% higher than conventional DRAMs. The WRAM has a 32-bit parallel interface and a 16-bit serial interface (for video). It also includes dual-color block write capability and some aligned block

VRAMs (SVRAMs) are available for dual-ported frame buffers. The Rambus DRAM (RDRAM™) represents a revolutionary approach to increasing bandwidth. RDRAMs transfer data at a 500MHz rate over a narrow, byte-wide bus, referred to as the Rambus Channel.

Extended Data Out (EDO)

EDO DRAMs are like conventional page mode DRAMs, except that the way in which data is disabled on a read is changed from the rising edge of CAS to WE. The outputs are held when CAS rises. This combined with a few other changes allow EDO DRAMs to cycle faster in page mode, therefore offering additional bandwidth. The signals RAS, CAS, WE, and OE remain the same as for a page mode DRAM, making the design transition to EDO straightforward. In order to provide sufficient bandwidth for graphics applications, the wide 16-bit I/O versions of these DRAMs are available in the 4Mbit generation. A 16Mbit page mode or EDO DRAM is not suitable for graphics due to granularity issues. Although a 16Mbit DRAM provides sufficient storage for a 2MByte frame buffer, it does not provide enough performance to meet associated display requirements.

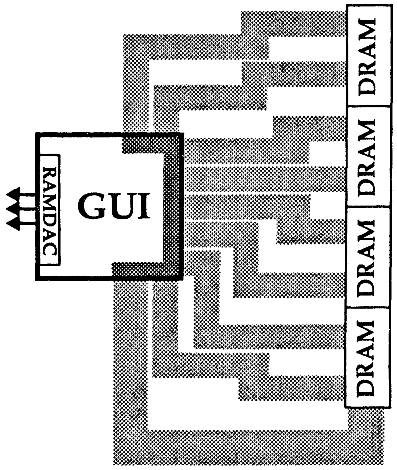
SDRAM and SGRAM

SDRAMs have an evolutionary design compared to conventional DRAMs. Internally, they are arranged in two banks, each independent and holding half of the DRAM bits. Available in a variety of bus widths (x4, x8, and x16), these devices are applicable for graphics at the 4Mbit (x16) and 8Mbit SGRAM (x32) densities. Although 16Mbit SDRAMs are becoming more available this year, the 16-bit wide bus is not fast enough to support the display refresh performance requirements of the majority of important display sizes.

While the interface to an SDRAM appears similar to a conventional DRAM, the timing and "commands" sent to a SDRAM are different from the RAS/CAS timing normally associated with a DRAM. The graphics controller designer must develop a new state-machine in order to support SDRAMs.

SGRAMs, which are based on SDRAMs, are offered in the 8Mbit density. They use a x32 interface in order to provide higher bandwidth for graphics. SGRAMs include the block write function, which allows SGRAMs to write as much as 32 bytes in parallel, but only every other clock cycle. They also have a single-color register, so that color expansion requires two passes for a font. Block write adds an increase in bandwidth of up to four times for pattern fills and up to two times for fonts. In addition to the wider bus, SGRAMs have one more pin, DSF, that is used to encode "commands" to the SGRAM.

Most SDRAM vendors offer parts that operate up to 66MHz. Data sheets have become available for SGRAMs, with some vendors showing 100MHz bin split parts. Since parts are not yet in production, yield on these parts has not been established. Achieving 100MHz operation on a board, using LVTTL signaling will be very difficult, because of board layout, clock/data trace, and skew issues. Memory expansion sockets further complicate the board design such that SGRAM configurations supporting expandable frame buffers are not expected to achieve the component's specified 100MHz operation in a system environment.



Single-Ported DRAM Approach

VRAM and WRAM

VRAMs are available in up to 4Mbit densities. The 4Mbit parts are arranged as x16 devices (for both parallel and serial ports) and are contained in 64-pin packages. Because of the their dual-port design and other features, VRAMs have cost up to twice that of single-ported DRAMs, and, thus, have been used only in high performance add-in cards. Most current card designs are moving away from VRAMs because of cost and the additional pin count incurred due to the second port.

Samsung offers the WRAM, a special version of an 8Mbit VRAM, which is intended to be priced 40%-50% higher than conventional DRAMs. The WRAM has a 32-bit parallel interface and a 16-bit serial interface (for video). It also includes dual-color block write capability and some aligned block

In PCI or VL based systems, most of the GUI controller pins are consumed by the local bus, frame buffer, and monitor interfaces. PCI requires about 45 signal pins, and VL uses even more. Counting power and ground connections, well over 55 controller pins are claimed by the local bus interface. Many of the current GUI controllers include an integrated RAMDAC to reduce overall component costs. In the integrated RAMDAC case, the monitor output requires an additional 10 to 15 pins. PCI requires the support of a BIOS ROM, which cannot sit electrically on the PCI interface and thus requires more pins. These factors leave about 120 to 140 pins of a 208-pin package unclaimed. Those pins are all that are available to support the frame buffer and any other functions.

Frame buffers using a 32-bit data bus to interface to single-ported DRAMs (EDO, SDRAM, SGRAM) will use 52 to 55 signal pins plus 12 to 18 power and ground pins for a total of 64 to 73 pins. A 64-bit data bus takes up somewhere between 105 and 115 pins. Therefore, a 64-bit bus uses almost all remaining pins on a 208-pin package. A 32-bit memory interface leaves about 45 pins for other controller functions, but its reduced frame buffer bandwidth limits the display resolutions that can be supported.

Dual-ported memory, such as VRAM or WRAM, contribute to higher subsystem costs in two ways: the dual-ported DRAMs are higher cost than single-ported DRAMs, and their interface can require more pins on the controller. In the past, the serial port was connected to a separate RAMDAC chip. As controller silicon has moved to smaller geometries and can accommodate more circuitry, many controller vendors have integrated the RAMDACs into their GUI controllers to reduce overall component costs. With the RAMDAC on the controller, the serial port must be connected back to the graphics controller. This situation further reduces the total number of available controller pins, or could force the controller into a larger, more expensive package.

Rambus DRAMs provide high bandwidth from the lowest pin-count interface. The interface to the byte-wide Rambus Channel takes only 31 pins on the controller. The interface consists of 15 active pins for data, control, and enable functions, with most of the rest power and grounds. The Rambus frame buffer saves up to 80 pins over alternative frame buffers. The pin-savings frees up the pad area on the controller die, allowing savings on controller die costs and controller package costs. The graphics designer is able to incorporate additional functions in the controller, such as support for video interfaces, feature connectors, or 3-D graphics support. The Rambus-based frame buffer allows the lowest cost controllers and lowest cost support for expanded feature sets.

Table 2: 2MByte Frame Buffer Comparisons

	EDO DRAM	4Mbit VRAM	WRAM	SDRAM	SGRAM	RDRAM
Organization	256K x 16	256K x 16	256K x 32	256K x 16	256K x 32	2Mx8 or x9
Number of Chips Required	4	4	2	4	2	1
Bandwidth Per Chip	80 MB/s	50 MB/s	160 MB/s	132 MB/s	264 - 400 MB/s	500 MB/s
Relative Cost	1.05	1.9	1.5	1.2	1.3-1.6	1.1 - 1.2
Board Area (component footprint only)	1.83 sq in., 1.36 sq in	.8 sq in	1.58 sq in	1.54 sq in.	1.1 sq in	0.1 sq in vert, 0.5 sq in hor
Package	40 SOJ or 40/44 TSOP	64 ZIP	120 PQFP	44/50 TSOP	100 TQFP	32 SVP or SHP
Pins Required on the controller	95 - 110	80 - 160	85 - 170	67 -72 or 114 - 119	68- 72 or 115-120	31

BOARD DESIGN CONSIDERATIONS

Because their electrical specifications influence board layout, PCI and VL bus interfaces often determine minimum board size. To ensure operation at up to 66MHz, the PCI and VL specifications dictate the location of the local bus interface and I/O connectors. The specifications also control the location of expansion connectors, such as the feature connector, VAFC, or Vesa Media Channel. PCI further specifies that no signal to the 32-bit interface portion of the controller can be more than 1.5 inches away from the connector, further restraining the controller chip location. PCI requires that only a single component can attach to the bus.

While the height of the board is generally determined by the PCI or VL specification, the board developer still tries to minimize the remaining sections of the board to keep costs low. The next factor to affect board size is the area required for the memory array of the frame buffer. As discussed above, PC designers have moved to supporting 64-bit data paths to multiple DRAMs connected in parallel in order to keep up with users' desires for higher display resolutions.

Due to its compact design, Rambus Technology helps to reduce board space. Most PC graphics frame buffers can be implemented with a single 8Mb or 16Mb RDRAM component. There are only 13 high-speed signals, plus about three other signals that need to be routed on the board. With the possible exception of SIn/SOut), all of the signals are routed as straight traces on the top of the board, leaving no signals on the bottom of the board underneath the memory array. The Rambus-based frame buffer can use only one to two square inches of board space, even if memory expansion is supported.

Smaller boards help to reduce the cost of the overall PC. The board size determines how many boards can fit on a single panel for PC board manufacture: the more cards per panel, the lower the PC cost.

Electro-Magnetic Interference (EMI)

All systems need to be able to pass FCC requirements, and, in many cases, more stringent TUV requirements. Good board and layout design practices are required to avoid coupling high-frequency noise to signals leaving the circuit board.

Although RDRAMs operate at a much higher frequency than conventional DRAMs, a Rambus frame buffer uses significantly fewer wires. The voltage swing on the Rambus Channel is much lower than the LVTTL signal swing. The Rambus Channel is terminated and very short (usually about one inch in most graphics designs). The result of these characteristics is that a Rambus subsystem radiates only 1/10th of the radiated energy that is generated by a comparable performance frame buffer using a 66MHz 64-bit wide data bus.

CONCLUSIONS

As silicon technology continues to advance, allowing more features and performance to be integrated into graphics controller products, PC users are demanding more and more sophisticated graphics and multimedia applications. The frame buffer implementation of the graphics card can be the single largest factor in determining the performance, cost, and board design of the end graphics subsystem. Today's graphics card designers are faced with many more feature options and DRAM alternatives. Out of all the frame buffer alternatives, Rambus Technology offers the highest bandwidth and potential for lowest system costs and expanded feature sets.

BIOGRAPHY

Billy Garrett is the Manager of Graphics Development at Rambus Inc. Rambus Inc. has developed the 500MHz Rambus Technology for high volume personal, portable, and multimedia systems. Mr. Garrett holds both a BSEE ('82) and MBA ('88) from the University of South Carolina. He has worked on a variety of design projects including: PC graphics boards, semi-custom ASICs for graphics, and main-memory applications, PC systems, UNIX servers, and X-Terminals. Mr. Garrett holds several patents in these areas.

TRADEMARKS

Rambus, RDRAM, RSocket, RModule are trademarks of Rambus Inc. All other brand and product names used may be trademarks of their respective companies.

Performance Requirements for Next Generation Chipsets

William E. (Eric) Mentzer
Marketing Manager
PCI Components Division
Intel Corp.
1900 Prairie City Rd.
Folsom, CA 95630

The technology acceleration represented by ever increasing demands for greater personal computer performance creates tremendous requirements for PCI chip makers. New generations of PCs, workstations, and servers will require tremendous flexibility as well as high performance.. For more information about Intel's PCIsets, call a local sales office or the Intel Literature Center at 800-548-4725 (in the U.S. and Canada).

PCI Bus Technology: Design Issues and Answers

Bernie Rosenthal
Vice President, System Interface Products
AMCC

In some circles,PCI local bus is described as a total system solution for increased performance in network adapters, RAID disk drives, full-motion video, graphics, and a wide range of other high speed peripherals. The features and benefits of PCI compared to buses such as ISA (Industry Standard Architecture) and EISA (Enhanced ISA) are many. Unfortunately, it's tough to get on the bus unless you've got somewhere between nine and 12 months of extra engineering time to invest. That's about how long it takes to run through the myriad of technical details and implement a custom solution, not to mention unravelling and resolving the ambiguities of the PCI bus specification.

AMCC's S5930-33 PCI Matchmaker Controllers address design issues associated with the PCI local bus and greatly reduce the time required to design a compliant adapter card. The controllers offer a flexible, general-purpose interface that easily connects most add-on applications to the PCI local bus. The devices provide a complete master/slave controller compliant with revision 2.1 of the PCI spec.

High Speed Data Acquisition System for Ultrasound Data Based on the PCI Bus

S.Freear, B.S.Hoyle, N.J.Bailey
Department of Electronic and Electrical Engineering
University of Leeds, LS2 9JT, UK.

ABSTRACT

Intravascular Ultrasound is an exciting new area of medical imaging. It is a pulse-echo technique employing high frequency transducers (20-30 MHz), this provides high resolution. The structures we are interested in visualizing are arteries, in particular the coronary arteries of the heart. The source of ultrasound is inserted into the artery. Cross sectional images can be built up by rotating the source. As arteries are not stationary objects, the capture, process and display has to be real time (40 frames/sec).

The aim of this project has been to develop a real-time clinical ultrasound imaging system capable of real time display and post processing. In capturing the raw ultrasound signal further processing can be applied to extract information on the nature of the tissue. Previously images were video processed and stored on video tape. A fully digital system does not suffer any degenerative effects, and a reliable database can be formed.

The first stage of the project has been to design a data acquisition card based on the PCI bus. The card accepts raw analogue ultrasound data. The data is digitally sampled at 250 million samples per second, and transferred across the PCI bus. The data is then processed to produce and image.

BACKGROUND

The source of ultrasound is at the end of a one meter long wire catheter, figure 1 below. The catheters are commercially available from Boston Scientific Corporation [Crowley *et al* 1989]. High definition cross sectional images of the artery provide important information regarding disease processes within the arterial wall [Gussenhoven, 1988].

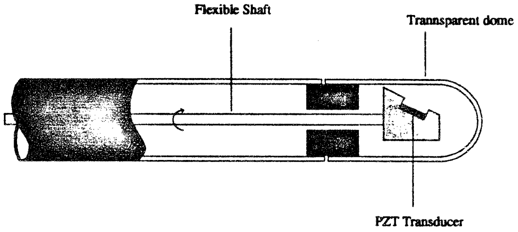


Figure 1 Cross section of an IVUS catheter

In this paper the design of an ultrasound data acquisition system is described. The essential difference from a commercial imaging machine (Bom, Lancee, Egmond, 1972) is that the raw ultrasound signal is being digitized before demodulation. Capturing the raw data enables the processing of valuable tissue characterization parameters (Recchia 1994) and the testing of a speckle reduction algorithm (Healey, Leeman 1993). The design is based around a standard Personal Computer, PC, and comprises three 'plug-in' circuit boards or cards; one to excite and receive the ultrasound signal, one to control the rotation of the catheter, and one to digitally capture the raw ultrasound signal.

DESIGN REQUIREMENTS

Pulser Receiver

The ultrasound generator/receiver must be capable of a maximum firing spike of 50 volts, and a half power pulse width of 4 ns. Maximum pulse repetition frequency will be 20 kHz.

Data Acquisition

Usually in ultrasound imaging systems digitization takes place after the signal has been demodulated. In order to extract additional information that can be used both for speckle reduction and characterization of the tissue structure the raw radio frequency (RF) signal is digitized. The signal has a bandwidth of 10 MHz and a central frequency of either 20 MHz or 30 MHz depending on the ultrasound catheter used.

Initially as data will be collected from phantoms and *in-vitro* tissue, real-time image display will not be necessary. In order to get up and running quickly with data collection the first phase of the design will therefore be a high speed acquisition system. Processing such as filtering and coordinate transformation both necessary for image display, can be performed in software. However these functions will be designed in hardware in the second phase.

Mechanics

The catheter comprises of a single ultrasound element and is therefore rotated in order to build up a 360º image. Maximum speed of rotation for a real-time image will be 2400 rpm. Accurate image display requires a constant angular velocity (0.1% error). The wire catheter is rotating within a sheath in the patients artery. Current monitoring must be incorporated within the motor drive in case the catheter becomes twisted.

General

Additional features required by clinicians would include simple image measuring tools, in order to calculate areas, frame by frame. Such features will be easily implemented in software. Rather than having a separate imaging system and post processing, an integrated system based on a personal computer, PC would be favorable. The design is modular and so can be based around a series of plug-in circuit boards or cards.

Data Rates

The motor control card and the pulser/ receiver are not data rate dependent as they both simply require initial setup. In order to produce a real-time image (40 frames/sec) a sustained data rate of 20 MBytes/sec. The ability to transfer the original data set will require even higher bandwidths (40 MBytes/sec)

HARDWARE DESIGN

Pulser/Receiver

The card to excite and receive ultrasound is based on an standard ISA bus plug in card. It is part of a flaw detector designed by AEA Sonomatic. The ultrasound firing voltage is variable from -5 to -200 volts in steps of 5 V. The width of the ultrasound spike is variable from 4 ns to 200 ms in steps of 5 ns. Both power and pulse shape are software controlled. Pulse repetition frequency is set from the digitizer card, and is variable to a maximum of 30 kHz. Safety features include shutdown in case of excessive current. This prevents incorrect software set-up.

Motor Control

The ultrasound element is rotated by a three phase brushless DC motor (Sonotron Ltd.). A drive board, also based on the ISA bus, contains all the circuitry to control the motor. The motor contain a shaft encoder which is fed to a phase detector in order to provide closed loop control. The phase detector output is summed with the reference phase output to produce an error voltage. This error voltage is integrated and used to modulate the motor drive voltage. If the rotor lags behind the stators magnetic field, a greater error occurs from the phase detector. This in turn increases the stator field and hence reduces the rotor lag. During power up, power down and freeze the motor is under board control so as not to damage the motor. Rotation of the motor is variable from 100 - 5000 rpm. To ensure patient safety the board is 'fail safe' and defaults to power down in the event of software error forcing the motor out of operating range. In case of excessive current the motor again shuts down, important in case the catheter has become twisted.

Digitizer

The high speed data acquisition card is based on a HI1166 from Harris Semiconductors. This part is a 250 MSPS flash A/D converter featuring differential and integral linearity of ±0.5 LSB or less, single power supply (-5.2V) and low power consumption 1.4 Watts. The digitizer card accepts the 'raw' un-filtered radio-frequency, RF, signal from the pulser/receiver card. The output of the A/D converter is demultiplexed by an ECL-TTL data latch (MC10H602 from Motorola). The four resulting data steams of 62.5 MBytes/sec is buffered in four 2K FIFO memory. In stage one of the design this FIFO memory is interfaced directly to a computer bus.

Bus Data Rates The maximum data rate on the AT ISA bus is typically 8.33 MBytes, assuming 16 bit data path, two bus clocks per transfer and an 8.33 MHz clock. The EISA bus is 32 bit wide , and, if the target device supports burst transfers, is capable of 33 MBytes/sec transfers. The Microchannel is capable of 40 MBytes/sec, being 32 bit and 10 MHz (Shanley, Anderson 1993). The Video Electronics Standards Association, VESA local bus solution produced a standard capable of 132 MBytes peak rate for burst reads. The Peripheral Component Interconnect (PCI) standard (PCI Local bus 1993) is also designed to operate at 132 MBytes (32 bit 33 MHz clock. The PCI solution backed by a Special Interest Group (SIG) of most major companies involved in PCs and is viewed as a more forward looking standard than VESA. The major benefit of the PCI bus is that it is processor independent and capable of higher sustained data rates than VESA, the PCI bus is becoming standard on high end PC's including Pentiums.

The design specification for PCI is currently based on revision 2.0 dated 30.4.93. Until mid way through 1994 there were few support chips that provided a simple interface to the PCI bus. This picture is changing with programmable logic (PLD) companies releasing code to implement an interface

PCI Interface The design solution chosen is based around the S5933 PCI controller from Applied Micro Circuits Corporation (AMCC 1994). The device provides address decoding, address sourcing, and burst transfers. The block diagram is shown in figure 2.

A 32 bit FIFO facilitates system to system synchronization, the EPROM interface allows pre-boot initialization. To aid prototyping applications a developers kit is available, based on the S5933 (PQFP). The kit comprises of a PCI compliant edge connector, two pre-programmed 22V10 PLD's provided for use in pass-thru mode. A serial NVRAM and byte-wide FLASH EEPROM are in system programmable though the NV build program.

FIFOs on board the S5933 are capable of bus mastering as each (read/write) has an address pointer and transfer count register associated with its PCI bus transaction. FIFO expansion is supported as shown in figure 3.

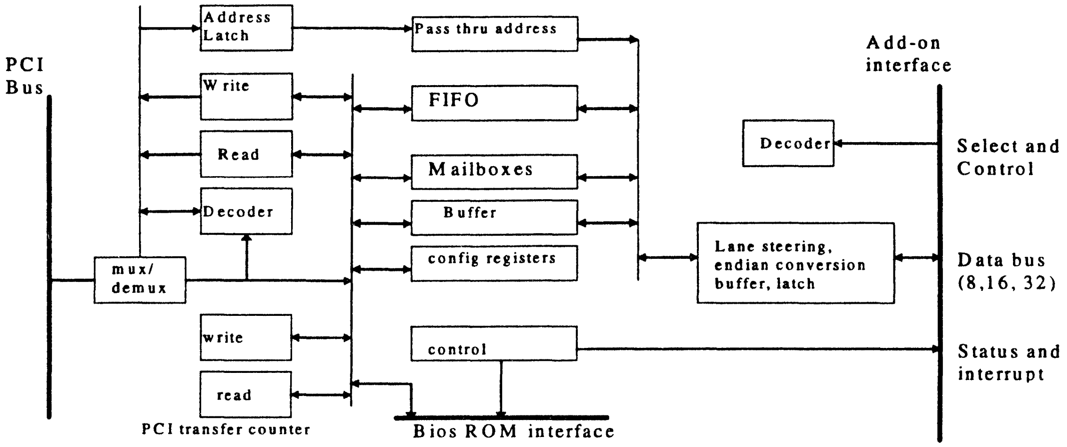


Figure 2 Block Diagram of the S5933 PCI Controller (AMCC 1994)

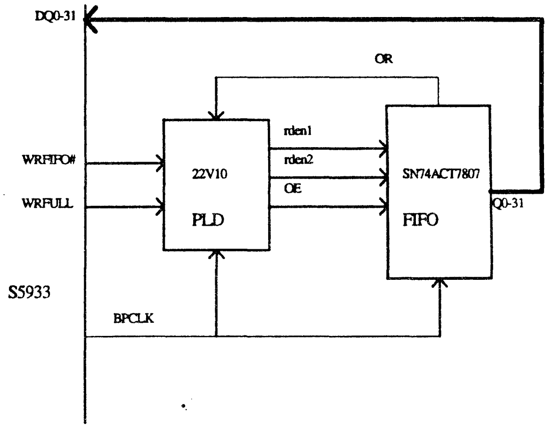


Figure 3 Block Diagram of External FIFO Interface

To support the high data rate (possible at a full system clock 33 MHz) a synchronous design is chosen. External FIFO control is provided by a 22V10 PLD. To read from the external FIFO the PLD generates read-enable (RDEN1, RDEN2) and output-enable (OE) and WRFIFO#. The PLD monitors the output ready (OR) flag of the external FIFO and WRFULL of the S5933.

PRELIMINARY RESULTS

The data acquisition system is capable of collecting 2.46 cm tissue depth of ultrasound data at 250 MSPS. This is using the full 8 KBytes of FIFO. The PCI controller transfers data across the PCI system bus concurrently with further acquisition. Currently a simple interface has been implemented displaying data in oscilloscope style. The major drawback with the system so far is that an image takes some time to compute (1 minute). Although this does not present a problem with *in-vitro* work, for the design to be used *in-vivo* a real time display is required.

CONCLUSION

The PCI bus on a modern computer is a powerful tool capable of the high speed transfer of data. For a device such as an ultrasound imager this has the benefit of a more flexible system. Data could be more easily stored and recalled digitally. The advantage of capturing the raw RF data enables the testing of algorithms designed to enhance the data set. The project is on-going, future work will aim at accelerating image display.

ACKNOWLEDGMENTS

The work is funded by the National Heart Research Fund. A charitable organization providing a lifeline for research into the cure and prevention of heart disease.

REFERENCES

AMCC, 1994. S5930-S5933 PCI Controllers. Data book .

Bom N, CT Lancee, FC Van Egmond, 1972. An ultrasonic intracardiac scanner.(US patent No. 1,402,192 filed February 22, 1973.) Ultrasonics;10:72.

Crowley RJ, PL von Behren, LA Couvillon Jr, DE Mai and JE Abele, 1989. Optimized ultrasound imaging catheters for use in the vascular system. *International Journal of Cardiac Imaging*, 4:pp145-151.

Gussenhoven EJ, CE Essed, CT Lancee *et al.* , 1988. Arterial wall characteristics determined by intravascular ultrasound imaging: an *in-vitro* study. *J.Am Coll Cardiol*, 11:22A.

Healey AJ, S Leeman, 1993. New approach towards speckle reduction. *International Conference on Acoustic Sensing and Imaging* conference proceedings pp 68-76.

PCI Local Bus Specification rev 2.0, 1993

Shanley T and D Anderson, PCI System Architecture, Mindshare Press, 1993 pp 5-6.

PCI Bus Performance
A Realistic Look

William G. Holland
IBM Networking Hardware Division, 3039 Cornwallis Dr.
Research Triangle Park, NC 27709
wholland@ralvm29.vnet.ibm.com

ABSTRACT

PCI Bus performance is advertised as "132 MB/s" (MBytes per Second). But, in measuring PCI busmaster adapters in available systems, aggregate throughput ranges from a low of 8 MB/s to a high of 85 MB/s.

As the lead designer of IBM's Auto LANStreamer PCI Token Ring LAN adapter, I am concerned with maximizing my customer's system throughput. Since this PCI adapter may be used in any PCI system, I, as the designer, can best satisfy the customer by understanding the available PCI systems and the interaction between various system designs and my product. Two questions will be explored in this paper:

- Where did all that PCI bandwidth go?
- How can my customers get maximum system performance?

I will focus on adapter and system design issues that are under the control of the adapter designer and/or the customer.

OVERVIEW

This paper starts with a look at how I view performance with regards to the PCI bus (within the context of this paper). A look at how the PCI adapter's design influences its efficiency on the PCI bus is second. Next, an overview of the factors that influence PCI performance for some of the most common PCI 486 and Pentium based systems is reviewed. A discussion of some specific alternative system designs, highlighting their performance benefits or penalties then follows. The last topic is a few suggestions for maximizing performance on current systems while awaiting systems that more fully exploit PCI's potential.

WHAT IS PCI PERFORMANCE

The PCI bus is most often characterized as a 132MB/s bus, but that is the maximum data transfer rate[1] and is only achievable by transferring infinitely large blocks of data. This is better referred to as the 'bandwidth' of PCI, a theoretical number. The data rate through the PCI bus (the PCI bus 'throughput') is determined by dividing the number of bytes transferred by the time taken to transfer them. I consider aggregate PCI-memory throughput as the key to PCI performance.

A high performance PCI device can be characterized as a 'master' or 'busmaster', transferring the bulk of the data directly to or from system memory; or as a 'target', 'slave', or 'shared RAM' in which the system processor transfers the bulk of the data between the device and system memory. Performance of a busmaster is determined by the PCI-memory interface. Performance of a slave device is dependent on more factors: the execution speed of the processor, the processor to memory interface, and the interface between the processor and the slave device. Each data item must be moved twice, once across the Processor-PCI interface, and once across the CPU-memory interface. My focus is on busmaster devices

Other Factors There are other factors affecting the performance of PCI systems and PCI devices. Here is a look at some other important performance parameters.

- For each PCI adapter, the portion of the PCI bus's total throughput that is available for its use is determined by how long it takes to start a PCI bus transaction ('bus access latency'[2]), and how long it is able to remain a master ('tenure') on the PCI bus. Bus access latency is affected by the system's arbitration logic, the PCI bus traffic in progress, and the adapter's own internal start-up delays (if any). An adapter's tenure on the bus can be limited by the PCI Latency Timer value set by the system, any target disconnect logic designed to limit data transfers, as well as data burst length limitations in the adapter itself.

- The PCI spec does not architect bus arbitration, so each system designer is free to choose there own arbitration scheme. PCI bus arbitration can be designed to efficiently distribute the available PCI throughput among the PCI devices, in proportion to the throughput that they need. The PCI Configuration registers 'MIN_GNT' and 'MAX_LAT'[3] provide sufficient information for a system to derive reasonable arbitration priorities and Latency Timer values. The commercially available systems that I have tested do not perform such optimization. Instead the arbitration is set to a fixed priority scheme, allowing the processor, or one of the PCI devices to hog the bus, potentially keeping other PCI devices locked out of memory for extended periods of time.

- Software design, and device drivers in particular, can greatly affect the performance of a PCI device as well as the overall system performance. A good hardware design should help the software by minimizing the overhead associated with

[1] PCI Local Bus Specification Revision 2.0 Section 1.5 pg. 5

[2] PCI Spec, Section 3.4.4, 3.4.4.1, 3.4.4.2, pp. 42-44

[3] PCI Spec, Section 6.2.4 pg. 159

data transfers (memory management and buffer manipulation required to support the device's busmastering) and processor-to-device interaction (interrupts and control I/O). Maintaining data buffers that are aligned with the 32-bit width of the PCI bus will increase the device's bus efficiency. Maintaining large contiguous buffers allows for longer data bursts and increased bus efficiency.

- The PCI bus will start to impact CPU performance if PCI devices are accessing memory when the CPU needs to access memory. It is important for a system to have sufficient memory throughput available for simultaneous processor execution and PCI bus memory access. Many systems do not have the processor-memory throughput to run at full speed (maximum MIPs) even without any other devices in the system. Moving significant data through the PCI bus can further restrict the processor's performance. These effects are difficult to detect on systems handling nominal amounts of data traffic. In network server systems and similar high-data traffic environments, this 'memory starvation' can be minimized by choosing a system with high memory throughput and choosing highly efficient networking and storage adapters.

- Processor cache size and design (write-back vs. write-through) can affect the percentage of processor memory transactions that can be handled within the cache. Every transaction handled within the cache is one less transaction that has to get through to memory. Write-back caches have an additional requirement that data writes to memory be checked against the cache so that processor and PCI data gets written to memory in the correct order.

Modeling all of these factors and designing systems for maximum performance are hot topics that, unfortunately, will not fit in this paper.

PCI ADAPTER EFFICIENCY

Maximum PCI adapter efficiency is achieved when the adapter causes no delays in data transfers beyond the architected PCI timings and the delays introduced by the system in which it is installed. The PCI architecture shows the shortest normal (non Fast Back-to-Back[4]) data transfer as requiring 3 clocks for a write, and 4 clocks for a read.
The individual clock cycles in a normal transfer are[5]:

- Write - Idle, Address, Data
- Read - Idle, Address, turn-around, Data

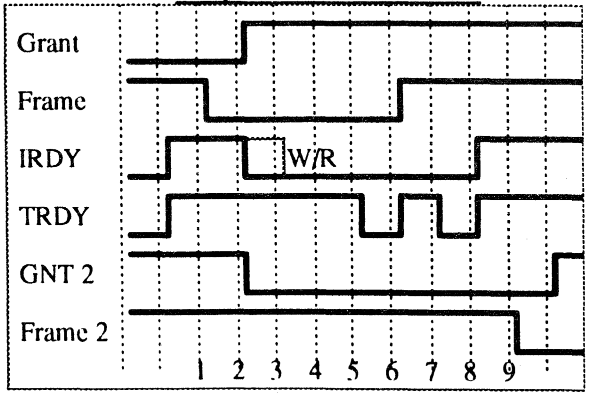
Most systems cannot provide this ideal 0-wait state timing. The delays can be summarized in two numbers. The first is the 'delay to first data', during which the PCI-memory bridge is receiving the PCI request, forwarding the request to the memory controller, reading or writing the data from/to system memory or an internal buffer, and then returning the first data phase on the PCI bus. For simplicity, I combine all of these delays into a single number calculated as the number of clocks from the prior Idle cycle (FRAME# and IRDY# not-asserted) to the first data transfer (IRDY# and TRDY# both asserted). The second number is the 'delay to subsequent data' which is the number of clocks from one data phase to the next during which the next memory or buffer location is being accessed and transferred to the PCI bus. The ideal read would be described as having 4 clocks to first data, and 1 clock to subsequent data. The ideal write would have 3, and 1. Borrowing from the nomenclature of cache and memory timing, I list the first 4 data phases, to show the relationship between subsequent data phases. Thus, the ideal read is '4-1-1-1', and the ideal write is '3-1-1-1'.

Typical delay to first data values range from 9 clocks (fast) to more than 15 clocks (slow). The 'delay to subsequent data' can range from 1 clock (ideal) to 4 clocks (slow).

In the sample transaction shown in the next figure, the system asserts Grant (GNT#) to the PCI device while a prior transaction is in progress. In this sample the master does not add any delays, the target adds 2 (read) or 3 (write) wait states to first data, and adds 1 wait state between first and second data (all by de-asserting TRDY#). To model this sample, add the architected cycles (3 for write or 4 for read) to the added wait states (3 for write or 2 for read). The total time to first data is 6 clocks. The time to subsequent data is 2 clocks (1 required, 1 added by the target). Thus the timing would be represented as 6-2-2-2.

Cycle 1 is the Idle cycle
Cycle 2 is the address phase
Cycle 3 would be the turn-around cycle for a read
Cycles 3, 4, and 5 are target wait states
Cycle 6 is the first data phase
Cycle 7 is a target wait state between data phases
Cycle 8 is the second (and last) data phase
Cycle 9 is the idle cycle for the next transaction

Sample PCI Bus Transaction



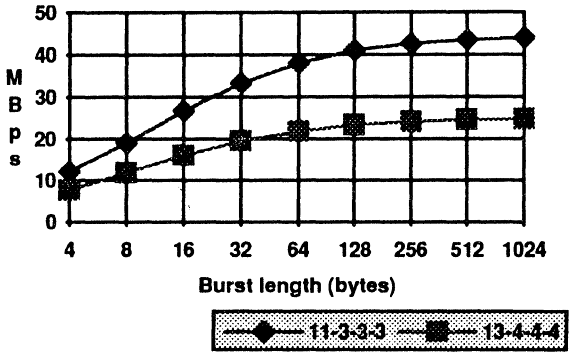
[4] PCI Spec, section 3.4.2 pg. 39
[5] PCI Spec, section 3.3.1 pp. 28, and 3.3.2 page 29

486 SYSTEM THROUGHPUT

Intel 80486 based systems appear to be the common PCI bus platform for the desktop. These systems are characterized by a single 32 bit wide memory data path to a 70nS memory, a small write-through cache, a fixed priority PCI bus arbitration, and a PCI clock frequency of 33Mhz. These factors combine to provide an aggregate PCI to system memory throughput between 30 and 40 MB/s.

The next chart shows theoretical throughput with timings representative of high and low performance 486-based systems. 11 clocks to first data (8 or 9 wait states) is faster than most 486 systems and combined with three clocks per data phase (2 wait states), would be a top performer among 486 systems. 13 clocks to first data with 4 clocks per data phase would be a slow system in this class.

486 PCI-Memory Throughput[6]



The PCI to memory interface is given the task of moving data between memory which is running at 70nS, and the PCI device which is running as fast as 30nS. Before memory can be accessed, the PCI request has to be given access to memory. This may involve waiting for the completion of prior PCI transactions within the buffers, processor transactions, or memory refresh. This plus request/response staging within the interface cause the additional delay to first data. On sustained bursts, the PCI bus must wait to allow the memory cycles to complete. This leads to an added wait state or two between subsequent data phases depending on the design of the memory controller.

Since the PCI bus and the memory are operating with the same 32-bit data width, each data phase on the PCI bus requires a memory access cycle. If the transactions do not use all four bytes on the bus, the effective data rate drops since fewer bytes are transferred in the same period of time.

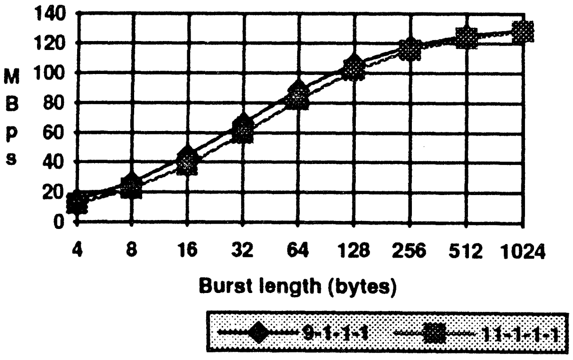
The processor-memory bandwidth on the fastest 486 (DX2-66, and DX4-100) systems is barely enough for the processor. Addition of very heavy I/O can be observed as a loss in processor performance. This would suggest that 486 based network servers or database engines should be configured so as not to overload the PCI bus with data transactions (creating an 'I/O bound' system). On systems with slower processors, the processor will generally determine the system overload point ('processor bound').

The IBM Auto LANStreamer PCI adapter is designed to burst longer on slower systems, providing for 196 or more bytes per bus transaction on many 486 systems. The measurements I have made[7] indicate that faster systems do achieve 40MB/s data throughput with this burst size. The slower systems I have measured were able to sustain about 30MB/s. Later I discuss system compromises that prevent achieving even this level of performance, either through limited or missing burst support, or inefficient arbitration design.

PENTIUM SYSTEM THROUGHPUT

Intel Pentium based systems are often the system of choice for processor intensive workstations, large network servers, and large database servers. These systems are differentiated by a wider (64 bit) memory path and often a larger write-back cache. P66 systems run the PCI bus at 33Mhz. For 33Mhz PCI bus systems, the aggregate PCI-memory throughput is between 75 and 85 MB/s, roughly twice the throughput of a 486 system.

Pentium PCI-Memory Throughput



The variation between high and low performance on Pentium systems is much narrower than on 486 systems. Most take a few less cycles to first data than their 486 counterparts. All Pentium systems I have examined provide subsequent data with

[6] The throughput graphs show calculated throughput numbers chosen to show the range of performance observed in systems of this class. The plotted performance does not reflect a specific system or set of measurements.

[7] All measurements quoted were made on IBM and non-IBM PCI systems available in the retail market as of January 1995. Various IBM and non-IBM adapters were studied in order to identify system behavior. All measurements quoted here were made with the IBM Auto LANStreamer Adapter. Throughput numbers were derived from logic analyzer tracing of the PCI bus transactions over the course of adapter buffer fill and purge transactions to system memory.

0-wait states. This provides a significant boost in throughput on burst transactions.

Current Pentium designs continue to use 70nS memory, so the time to first data is not much better than 486 timings. But the 64-bit wide memory allows the memory interface to stay ahead of the 32-bit wide PCI bus once the transaction is started. This is a significant improvement even when transferring as few as 16 bytes (at that point, throughput already equals the 486 system's best throughput.) If some of the 8 bytes available on the 64-bit memory bus are not used by the PCI device that portion of the available throughput will be wasted.

Many Pentium systems do not operate at a multiple of 33Mhz. In most P60 and P90 systems, the PCI bus runs at only 30Mhz, a 10% PCI throughput reduction over an equivalent 33Mhz system. Some newer processor/memory-PCI bridge chip designs allow the PCI clock to be asynchronous (other than an even multiple of the processor/memory clock). Such systems should be examined closely, since asynchronous interfaces can add a clock or two delay to each data transaction. This could negate the increase in PCI bus speed.

Pentium systems are usually limited by their memory throughput. Inefficient adapters moving large amounts of I/O can impact the processor performance. System configurations should be designed to properly balance the I/O and processor requirements of the application.

The IBM Auto LANStreamer PCI Adapter is designed for 0-wait state bursts averaging greater than 64 bytes each. The measurements I have made indicate that on Pentium systems the typical burst is 76 to 88 bytes, and throughput ranges between 75 and 85 MB/s. Some Pentium systems will Target Disconnect on bursts of this length, and therefore place a cap on PCI bus throughput, often below 70MB/s. This is discussed in the next section.

At 80MB/s the IBM Auto LANStreamer PCI adapter would use less than 3% of the PCI bus. The CPU utilization during high LAN traffic is 10% or less on most Pentium processors. The typical server system could readily support more LAN connections than there are available PCI slots. Lab testing with three of these LAN adapters shows no noticeable degradation in network throughput (neither the PCI bus nor the CPU is overloaded.) while servicing 3 separate network segments.

LOW-COST SYSTEM COMPROMISES

To provide lower cost PCI bus systems, various design compromises have been made which, while attempting to save money, sacrifice PCI bus bandwidth.
The first one is to limit the data buffering capabilities in the processor/memory-PCI bridge chip. This will be seen by a PCI busmaster device as an inability to do long bursts and/or as additional waits states while bursting.

- One design point that appears to be popular is for memory to issue a target disconnect[8] whenever a cache line boundary is crossed. Thus the maximum burst size is 16 bytes on a 486, or 32 bytes on a Pentium. But, on transactions starting in the middle of a cache line, the burst size will be even less. This can limit total PCI throughput to 15-25MB/s (486) or 60-65MB/s (Pentium). Some chip designs are less severe, disconnecting at 2KB or 4KB (page) boundaries. The impact to average burst size is less, since only 1-in-16 to 1-in-64 of the bursts will be cut short due to the target disconnect.

- Some chips do not support burst mode at all, issuing a target disconnect with the first data phase of every transaction. Although this was probably not the intended design, the compromise was that such a chip is 'good enough to ship'. In a 486 system without any data burst capability, the maximum theoretical PCI throughput would be 19MB/s. Of course, as luck would have it, the system I measured (which shall remain nameless along with all of the systems discussed in this paper) had additional compromises in the arbitration and PCI bus access parts of the time to first data. The net result was an aggregate PCI throughput of just over 8MB/s. For reference, ISA could theoretically deliver over 5MB/s![9] The IBM Auto LANStreamer PCI is still able to run well in such a system, and multiple adapters will operate properly in a lightly loaded network environments. But, with increasing network traffic, the PCI bus quickly saturates and the system bogs down.

- In balancing processor performance with PCI busmaster device performance, the system must have a way to choose which one will get priority on accesses to memory. Some systems do not assure that a PCI device, once granted the PCI bus, will get fair access to memory. This results in PCI bus transactions thrashing on retry from the memory interface during times of high processor-memory transactions. In a networking environment this means the processor doing background tasks like memory clean-up can lockout the network adapter long enough to loose frames. Network protocols will generally handle this situation without impact, but if the frequency of network adapter lockout is high enough, the network connection will drop, or the network will flood with retries.

- Most current PCI systems implement a fixed arbitration scheme. That means that the device at the bottom of the priority list will be locked out of the PCI bus when any other devices is using the bus. Neither the PCI device nor its associated software has a way to determine the arbitration scheme in use, or the adapters relative priority. Only the system designer knows how the arbitration will affect a specific adapter's performance.

[8] PCI Spec, section 3.3.2, pp 33-36

[9] 2 bytes every 3 B-clocks (120nS each) = 2/360nS

- The use of PCI-to-PCI bridge chips (to add additional PCI devices or slots to a system for example) will add a small amount of overhead to transactions crossing the bridge chip. For highly efficient devices, the added overhead should be negligible. But, any inefficient PCI bus transactions will become more inefficient when passing through an additional bridge.

ADVANCED SYSTEM DESIGNS

At the high end of the price grid, systems are becoming available that provide higher memory bandwidth, and/or higher PCI bandwidth.

- Within the current PCI architecture are some extended commands[10] that will allow the system to handle memory requests more efficiently. Memory Read Line, and Memory Read Multiple both allow the system to 'know' how much read pre-fetching to do on behalf of the requesting device (Because these commands imply an intent to request more data beyond what the system would otherwise assume). Memory Write and Invalidate (MW&I) can speed up writing data to system memory by skipping the cache snoop that has to be done to guarantee that system memory is current prior to the PCI data getting to memory. This is particularly beneficial with a write-back cache where cache snoop accesses penalize memory throughput. The IBM Auto LANStreamer PCI Adapter implements these functions, along with the logic to dynamically choose the best command to use for each transaction. The first systems that will see a performance benefit with the extended commands should be reaching the retail market in the first quarter of '95. In multiple adapter configurations utilizing one or more processors with write-back cache(s), there should be a measurable improvement in overall system memory throughput.

- Fast Back-to-Back transactions were ignored throughout this paper, but they can remove one clock to first data when the busmaster is performing multiple transactions during its tenure on the bus. Some systems today disable the Fast B2B feature (others should - since they fail to run if Fast B2B cycles are issued.) Many systems already honor this type of transaction, and receive a small performance improvement for their efforts.
- To get additional benefit on those systems that do allow Fast B2B cycles, the arbitration logic would need to keep GNT# asserted to the device so that it could issue additional transactions. Some of the current arbiter designs only issue GNT# long enough to start one transaction. Then, GNT# is issued again if the device continues to request the bus. But often GNT# is not present in the right cycle to allow Fast B2B.

- Some second generation PCI devices are showing up with double or more data buffering than the original versions. This can allow for better performance due to the longer burst size, but I suspect that it is actually a reaction to the 'memory starvation'/'memory lockout' related problems. The more buffer space a device has to work with, the more likely it will survive an occasional lock-out from memory. So, the buffer thresholds are set to minimize starvation, rather than maximize burst length. Some products can get better throughput through software parameters or design (buffer size, buffer thresholds, starvation recovery procedures, etc.)

- The next step for system designs is to double the memory speed to 66Mhz. Time to first data will not get significantly shorter, since 66Mhz memory is often just interleaved 70nS memory. Since Pentium systems already run w/0-wait state between data phases, there will be no direct PCI throughput increase. The benefit lies in the fact that the data for the PCI device can be transferred between memory and a buffer in half the time it takes the PCI device to transfer the same data. During the other half of the time, memory is available for the processor to access memory. So, data may move on the PCI bus at about the same rate, but the processor and PCI devices no longer have to contend for memory access.

- Other design possibilities include using faster memory chips 60nS, 50nS...20nS? The cost will keep this restricted to the most demanding applications for a few years at least. But the delay to first data could be cut significantly, in addition to reducing processor/PCI memory contention.

- The PCI bus itself is enabled to double or quadruple its throughput. The ability to implement a 64-bit wide PCI data path exists today, and the double speed (66Mhz) PCI chips are coming. One limitation on both of these is that the Master and Target must both provide the new feature for it to be a benefit. And, in addition, with the 66Mhz option, all devices on the bus must be able to run at 66Mhz, or none can.

- Some processor/memory-PCI bridge chips will provide significantly more buffering than current chips, more like an additional level of cache. This will effectively increase the system memory bandwidth for PCI devices the same way the L1 and L2 caches benefit the processor.

- Providing multiple memory controllers for different regions of memory, and separate PCI busses for small groups of devices can be combined to segment the PCI transactions across different interfaces, each of which only needs to support a portion of the total system throughput.

- Adding memory-PCI bus interfaces (multiple Host-PCI bridges) can add additional PCI slots to a system while improving the throughput to memory, since each set of PCI slots has its own memory access.

[10] PCI Spec, section 3.1.1, 3.1.2; pp 19-22

PCI SYSTEM PERFORMANCE TUNING

Some vendors, including IBM, have software methods to work around, or fix, specific problems on some specific chips. IBM has chosen to deliver these 'system tweaks' in the form of a program that comes with the adapter and is run after the system is booted, but prior to adapter initialization. This program will search PCI configuration space, determine which chipset is in use in the system, and then apply corrections as needed to assure that the system operates properly with the Auto LANStreamer PCI adapter. Look for such a program from your adapter vendor, or check the documentation for their recommendations.

The more data traffic that is re-moved from the subordinate bus (ISA/EISA/MC) and placed on the PCI bus, the more efficiently the PCI bus can operate. ISA disk, LAN, or video adapters can steal a large portion of the PCI bus due to the slow transactions on ISA. EISA and Micro Channel subordinate bus devices are certainly better than ISA, and come pretty close to matching PCI performance under ideal conditions. But in general, a native PCI adapter will be more efficient that a comparable device on the subordinate bus. Under non-ideal conditions (high bus utilization, or shorter burst lengths) the difference between the bus types becomes more pronounced. In addition, if you have a non-bursting EISA adapter or a non-streaming Micro Channel adapter, a busmaster PCI device should be more efficient.

On systems in which the BIOS allows direct access to the chip level registers, look for these parameters and see if the settings match those shown here: [11]

Arbitration = Fair
Arbitration priority =
PCI devices, EISA/ISA/MC devices, then processor
Burst mode = On
Latency Timer = 60
PCI Line Buffers = On
Posted PCI Write Buffer = On
Posted CPU Write Buffer = Off
CPU-PCI Write Posting = Off

CONCLUSIONS

I am very excited that new systems coming out at the high end that may actually deliver >100MB/s to the PCI bus. At the same time, I am concerned for the future of PCI when I see new 'low cost' chipsets that cut performance to 1/4 or less of what should be attainable.

On the adapter side, many vendors have long ago figured out how to pass the PCI Compliance Tests, test their adapter, and deliver it to their customers; and now they are delivering second and third generation PCI products that really begin to exploit the PCI architecture.

The PCI bus is quickly becoming the bus of choice on many non-x86 processor platforms. The information presented here is equally applicable to other types of PCI systems. The key is to match systems based on the available PCI-memory throughput. That gives you a starting point for estimating PCI device performance. Then you can start looking at how specific PCI hardware behaves in specific systems. Software issues and processor-device communications will tend be very processor specific. PCI bus transactions on the PCI bus in IBM's PowerPC reference platform (a prototype system with a 601 processor) look just like PCI transactions on a comparable x86 based systems.

The PCI architecture enables some very powerful system designs. Through careful analysis and design, some products will become the 'best of breed', while others will be well positioned to be the 'low cost solution', and yet others will find their way to oblivion because they had one of the million possible combinations of high cost and poor features.

BIOGRAPHY

Bill Holland is currently the Busmaster Adapter Technical Team Leader in Token Ring Adapter Development. The team's latest product - The IBM Auto LANStreamer PCI Adapter (a PCI Token Ring network adapter) started shipping 4Q94. Beyond the Token Ring group, Bill is the IBM Networking Hardware Division's technical focal point for PCI product development.

In his 11 years with IBM, Bill has had assignments at 4 sites, including: Circuit Board Manufacturing Process Development, Mainframe (3090) Processor Design, Large Scale Computing Division Product Engineering Manager, Joint Customer Study Liaison during development of the parallel CMOS 'mainframe' computers.

[11] **WARNING**: Some systems have bugs that are fixed by very specific settings of the chip registers, usually applied by the system BIOS. Changing these settings could cause the system to operate incorrectly. Keep track of prior settings for all parameters. Test any alterations in a test environment. Always solicit your system vendor's recommendations. When you start manually altering hardware registers, data can be lost or corrupted in memory, on disk, and even across a network.

MPEG Decoders on PCI

Shrikant Acharya[1] and John E. Crosbie[2]
Margi Systems Inc.
702 Topawa Drive
Fremont, CA 94539
Ph. (510) 657-4435 Fax (510) 657-4430

Abstract

Fast communications are necessary in data intensive operations such as real-time video, sophisticated graphics, local-area networks etc. and these are all benefited by the advent of local buses e.g. PCI and VESA local bus. The paper will review the bandwidth requirements of the MPEG-I and MPEG-II and the aspects of implementing MPEG-I and MPEG-II compressed video standard decoders on PCI buses. The coming to the market of full feature length movies on CD-ROMs has ushered the market for MPEG-I/MPEGII technology in a big way.

1.0 Video Compression Standards

1.1 MPEG-I Digital Audio/Video Compression Standard

The MPEG-I standard prescribes that all standard decoders should be able to decode MPEG System bit streams coded at 1.5 Mbits/second. The encoded image shall be non interlaced and confirm to the SIF image size (352x240 at 30 frames/second for NTSC while 352x288 at 25 frames/second for the PAL system). The standard provides transcoding between motion picture rates of 24 frames/second , PAL at 25 frames/second and NTSC at 30 frames/second. The MPEG-I bit stream was designed to be compatible with the bit rate supported by the single speed CD-ROMS in existence at the time of the formulation of the standard. Single speed CD-ROMs provide a standard bit stream at 150 Kbytes/second. However, the MPEG-I algorithm can accommodate data rates of up to 5 Mbits/second.

1.2 MPEG-II Digital Audio/Video Compression Standard

As MPEG-I could not meet the quality and display requirements of the broadcast industry, a higher data rate audio/video compression standard called MPEG-II was proposed. The MPEG-II standard provides for the coding of bitstreams from 5 Mbits/second to 10 Mbits/second. This standard is also more flexible in accommodating different audio encoding standards for incorporation into its system. Currently the two audio schemes are the 1) MPEG-I audio standard (MUSICAM) and 2) Dolby AC-3 audio compression algorithm. The image size is CCIR-601 i.e., 704x480 at 30 frames/second for NTSC, 704x576 at 25 frames/second for PAL The MPEG-II standard also proves for field to field motion compensation (a technique for reducing frame to frame or field to field correlation between images and results in reducing the effective bit rate of the compressed bit stream) as opposed to frame to frame motion compensation in MPEG-I. This provides for a better correlation between frames and leads to higher picture quality at a given bit rate.

One of the offshoot implementation of the MPEG-II standard is being adopted by two rival high density compact disk formats. They are DVD (Digital Video Disk) and High Density Compact Disk (HDCD).

1.3 DVD vs. HDCD

The Digital Video Disk (DVD) proposed by the consortium of Time Warner, Matsushita and Pioneer is compact disk with 10 Gigabytes capacity dual sided and plays movies up to 270

[1] Shrikant Acharya is President of MARGI Systems, Inc
[2] John Crosbie is VP of Engineering at MARGI Systems, Inc.

minutes long (135 minutes to a side). The rival standard by SONY and Philips is HDCD with a capacity of 7.4 Gigabytes capacity on dual sided compact disks as in DVD. MGM/UA and other Hollywood studios have given their backing to DVD ostensibly for its higher density over rival HDCD. Both these standards will use the MPEG-2 video encoding compression algorithm and use the Dolby AC-3 encoding technique for the audio. (Dolby AC-3 is likely to be adopted by the MPEG committee as one of the audio encoding mechanisms) for their compact disks. Even though Hollywood has chosen DVD, it does seem that with SONY and Philips with their prior entrenchment in the PC area could influence the market to use HDCD thereby splitting the market between Hollywood studios on one side (DVD) and PC manufacturers on the other side (HDCD).

The DVD system's peak data rate is 10 Mbits/second. It sustains an average transfer rate of 4.94 Mbits/second for the full 135 minutes on each side.
Other DVD parameters are:

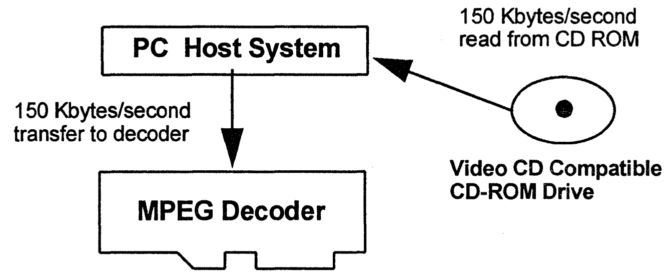
Disk diameter	120mm(5 inches)
Disk thickness	1.2mm (back-to-back bonding of two 0.6 mm thick double sided)
Memory Capacity	5 Gigabytes/single-side, 10 Gigabytes/double-sided
Track Pitch	0.725 micrometer
Wavelength of laser diode	650 nanometers
Numerical aperture	0.6
Error Correction	RS-PC (Reed-Solomon Product Code)
Playing Time	
Movies	135 minutes/side, 270 minutes on both sides (at an average data rate of 4.94 Mbits/second for image and S sound.
Secondary use of broadcast programs	74 minutes/side, 148 minutes on both sides (at an average of 9 bits/second for image and sound)

2.0 Need For PCI

Compressed data bandwidth as a measure of the total bandwidth for the ISA bus (real maximum transfer rate is 2 Mbytes/second) is 15% (.3/2), while it is only 1.5% (.3/20) of the PCI band width (typical transfer rate of 20 Mbytes/second).

As the pixel data rate is much higher then 2 Mbytes/second (shown below) even at SIF display size, all ISA based systems route pixel data out of separate video bus on the decoder board. It manifests as a ribbon cable brought out of the board to be connected to a video display board.

The figure 1. indicates typical bandwidth taken when data is transfered from a CD-ROM to a MPEG decoder board on the PC bus.



Total transfer badwidth is 300 Kbytes/second. First read through the ISA bus from CD-ROM and then write data into decoder through the ISA bus.

Figure 1. Typical Video Data transaction on the ISA Bus

The numbers below project the data rates required to display images of a particular size and display rate.

For SIF Images (352x240 for NTSC and 352x288 PAL)
Frame rate 30 frames per second or 60 fields per second

NTSC Date rate = 30 (frames/second) * 352x240 pixels *2 byte/pixel
= 5.0688 MBytes/second

PAL Data rate = 25 * 352x288 *2
= 5.0688 MBytes/second

For CCIR-601 Images the data rate is:

NTSC Data rate = 30 * 704*480 * 2
= 20.2752 MBytes/second

PAL Data rate = 25 * 704*576 *2
= 20.2752 Mbytes/second

At 60 fields per seconds each of the data rates indicated above shall double.

With the indicated data rates even PCI bus bandwidth can be exhausted. Therefore unless the PCI bus throughput is made substantially larger than 20 Mbytes (PCI bandwidth can go up to 132 Mbytes/second) it is not advisable to route the pixel data through the PCI bus into a video card. However, the compressed data rate occupies less than 2% of the PCI data rate and therefore the remaining bandwidth can easily be used to enhance the response of the Windows system, get data off the network, or perform sophisticated graphical operations.

In MPEG-II the data rates coming from the CD-ROM is going to be between 5-10 Mbits/second. Effectively as indicated for ISA calculations in Figure 1., the bit rate will be 10-20 Mbits/second on the bus. At 10 Mbits/second the MPEG-II bit stream shall occupy 60% of the ISA bus bandwidth but only 6% of the PCI bandwidth. The advantages of using PCI for MPEG-II decoder designs are clearly evident.

2.1 PCI Bus

The PCI bus can be clocked at the maximum clock rate of 33 MHz. PCI employees a 32 bit multiplexed address and data bus to give an effective data transfer rate of 132 Mbytes/second (a 64 bit data bus is optional). The PCI bus is an unterminated transmission line having CMOS loads. An unterminated bus uses reflective waves instead of incident waves to transfer data. The "reflected-wave" switching, creates a voltage wave that travels down the transmission line which is not enough to cause a logic transition. It must wait to be reinforced by the wave reflecting off the end of the transmission line. PCI defines the worst case signal propagation delay of 10 nano seconds.

In contrast the ISA bus is an incident wave bus, the maximum clock rate is 6-8 MHz and the data path is only 16 bits wide. The maximum transfer rate is around 5 Mbytes/second, but realistically that bandwidth rarely exceeds 2 Mbytes per second. The applications on the ISA bus will however will not disappear any time soon as it has the huge installed base.

2.2 PCI Design Implementations

A number of companies have started providing PCI interface controllers for various applications e.g., IDE controllers, Ethernet controllers, VGA controllers. Manufacturers have started to incorporate PCI interface into their base silicon design, obviating the need for a special purpose PCI interface chip. NCR offers 53C810 and 53C85 PCI-to-SCSI processors that connect directly to the PCI bus. Future domain also offers a single-chip SCSI-2 interface to the PCI bus. Adaptec AIC-7870 is a similar PCI-SCSI interface chip. Symphony Labs SL82C101P provides a PCI-IDE interface with ability to support up to 4 drives and offers IDE primary and secondary addresses selection and an automatic stand by mode for power saving. The PCI-IDE interface chip from Opti 82C621 from Opti Inc. also supports four IDE drivers and contains 16-byte read prefetch and FIFO. Cirrus Logic and Tseng Labs have their new VGA controllers that already incorporate the PCI designs inside their chip.

However, there are many applications and prototype designs, including the MPEG-I and MPEG-II decoder designs which may require the use of off the shelf general purpose PCI controllers or the use of FPGA's that can effectively implement an independent PCI interface.

Applied Micro Circuits Corp. (AMCC) offers the S593X series of general purpose PCI bus controllers. LSI Logic provides PCI-32 FLexCore building blocks. Specialized FPGA's that can be adapted to service PCI design purposes are available from Altera (MAX 7000/8000, FlexLogic 8160). AT&T's Optimized Re-configurable Cell Array (ORCA) family of .5 micron CMOS FPGAs offer 12K-26K gates and as many as 384 usable I/O pins. Xilinx offers XC7300 family of EPLDs. Xilinx even guarantees PCI compliance with their XC73108-10 EPLD. In a typical design XC71308 functions as the PCI bus interface, an XC7354 functions as the error handler, and an XC3190A functions as a memory and back-end controller.

2.3 Layout considerations

The PCI specification 2.0 gives special mention for the placement of the control and input multiplexed address and data. This is to guarantee the bus master (host) a load on each line that is within the specifications for line capacitance and load. To minimize transmission line effects and input capacitance, signal stubs should be kept to a minimum. The maximum trace length for 32-bit signals is 1.5 inches. Additional signals are limited to 2 inches, with the exception of the clock which is exactly 2.5 inches. The designed is advised to review the suggested pin out for pin placements before completing a PCI design.

Do not daisy chain clock lines. All devices on the PCI bus should have the same exact distance from the master clock output. The PCI clock source has to arrive at each device with a maximum skew delay of 2 nano-seconds. Therefore the user should buffer the PCI clock coming into the controller instead of distributing it on the board.

2.4 Specific Issues related to the Use of EPLDs

Designers have to be beware about being carried away with the assertion of ASIC manufacturers that their part is very flexible. Designers in their haste to get a prototype done overlook the fact that their design could potentially occupy more than 75% of the macro-cells in their FPGA. This can make the design very inflexible to routing. Especially the problem gets exacerbated if without configuring and compiling the design, pin assignments are already made just because of the availability of the I/O pins. A case in point is the FLEX logic 8160. Intel's application note indicates a 90% occupancy of the macro cells for target implementation. This can make the part very difficult to route. Designers are advised to first compile the design and check where the fitter is able to place the PCI sensitive pins. The part should be placed in the schematic with pins only after the designer is nearly complete with his design and has left room only for minor modifications. Otherwise the cost in lost time is enormous.

3.0 PCI MPEG Design

3.1 Hardware Design

Generic designs for the MPEG-I decoder are presented to the reader. This design can be extended to incorporate the MPEG-II decoders as well. MPEG-I decoders are available from C-Cube Microsystems, Zoran Semiconductor Inc. and Thomson CSF. More companies may bring their MPEG-I designs to the market in 1995, but the above three companies have had an established presence. In the case of MPEG-II decoders, only LSI-Logic and C-Cube Microsystems are prominent in their offerings.

Listed below are two typical design examples for the MPEG-I playback systems. In Figure 2. the output is to a VGA screen. This example uses the new generation of VGA controllers that provide on board PCI interface control and the ability to accept pixel data and also control the size and placement of the window.

The second example suggests a possible interface to a Video Windowing environment. This example assumes that the board has a separate VGA controller. This design provides video using the chroma key feature. The output DAC as shown in the figure combines the VGA from the feature connector with the pixel data coming from the Video Windowing chip.

Both the above examples can enhance their throughput performance by using a bus master design instead of a target PCI design. A Bus master can initiate the transfer of the MPEG data from the CD-ROM driver without the CPU intervention. This adds more complexity to the PCI card for MPEG-II but also increases the efficiency of data transfer on the bus.

3.2 Software Design

The software should be structured into layered modules for ease of integration and maintenance. The base module will have the lower lower level primitives that provide access to the hardware, provide setup of the hardware and basic data throughput in and out of the hardware. The next level of structure is the high level primitives that will provide more macro functions as opening buffers and manipulating buffers and providing rate control to the hardware. The top level of the driver will have the MCI commands that will open a file, close a file, play the file etc. A complete software package for DOS and Windows environment should include the following:

3.2 Software Design

The software should be structured into layered modules for ease of integration and maintenance. The base module will have the lower lower level primitives that provide access to the hardware, provide setup of the hardware and basic data throughput in and out of the hardware. The next level of structure is the high level primitives that will provide more macro functions as opening buffers and manipulating buffers and providing rate control to the hardware. The top level of the driver will have the MCI commands that will open a file, close a file, play the file etc. A complete software package for DOS and Windows environment should include the following:

3.2.1 Digital Video MCI (Media Command Interface) Driver

The new set of drivers for the Windows environment should follow the guidelines setup by the OM-1 (Open MPEG Committee) and this also includes MCI Overlay functions. The overlay functions provide the ability to overlay video over the VGA window. Recently Philips Interactive Media has started bringing movie titles on Compact Disk using MPEG-1 technology. Philips has enveloped it into a specification known as the Video CD. The MCI driver should provide support for playing Video CD titles.

3.2.2 OM-1 DOS Driver

The OM-1 DOS driver specification is also published by the OM-1 committee. This is ostensibly to provide a uniform standard for the Video Game writers to follow in designing their games. The OM-1 DOS specification will provide a standard that all manufacturers of MPEG-I decoder software and hardware can follow. The software consists of a TSR (Terminate and stay resident) driver which can intercept OM-1 commands and play back MPEG system files. The TSR driver should also include Video CD support.

The design of the above software drivers have the following sub sections:

1) **Windows MCI Overlay Functions**
The MCI overlay commands will rely on the DLL (Dynamic link library) provided by the manufacturers of various Video Windowing chip sets (e.g., Auravision, Pixel Semiconductor). The DLL follows the MCI interface. Any MCI software designed for the MPEG decoder shall issue commands to the Video Window DLL . While the MPEG decoder provide pixel data output, the Video Windowing section provide the ability to do color conversion, window positioning, window sizing etc.

2) **Develop Basic MCI Digital Video Driver**
It is important to have a basic software in place to test and debug the hardware as soon as the hardware is available. This software is a very basic MCI driver with the capability of Open and Play. This driver performs the following tasks:

- Basic Windows Installable Driver structure, Driver Setup.
- Hardware initialization of the Audio and Video Decoders.
- Open and parse MPEG Bitstream stored in a DOS file.
- Delivery of MPEG Bitstream to Decoder.

This driver will operate with the MCITEST program. Driver setup includes the ability to set the hardware I/O Port, Interrupt Channel and DMA Channel. A hardware presence detect can be performed to determine if the board is installed and configured correctly.

Delivery of MPEG data to the Audio and Video decoders requires the following tasks to be performed:

A critical component of a successful Audio/Video computer system is the data path design. The different data rates of the source file and the destination decoder must be accounted for by a number of bitstream buffers. Optimum hard drive and CD drive performance is obtained by using a particular read block size. Hard Drive data rate interruptions due to thermal recalibration and, in the Windows environment, data access by other applications make a large rate buffer necessary. A buffer of 1 second of data should be sufficient.

3) **Completion of Windows MCI Digital Video driver**
The MCI software should be capable of operation with Media Player (a standard Windows application to play video files). This driver will support MPEG Bitstreams stored in a DOS File only.

The MCI commands that should be supported for OM1 compatibility follow. A description of the function of each command can be found in the Microsoft documents: "Microsoft Multimedia Standards Update, MPEG Command Set for MCI" and "Digital Video Command Set for the Media Control Interface" (Microsoft part# 098-37538).

Configuration/Status commands
Configure, Info, Signal Set, SetAudio, SetVideo, Status
Windows Commands
CreateVideoDC, UpdateVideoDC, ReleaseVideoDC, Update, Where Window
Clipboard commands
Copy - Copy specified frame to clipboard as 24bit BMP
Player Control Commands
Freeze, Unfreeze, Pause, Put, Resume, Seek, Step, Stop

4) **DOS OM1 driver and DOS MPEG Player**
The OM1 DOS Driver is a Terminate Stay Resident (TSR) program that provides similar functions to the Windows MCI Driver. The functionality of this driver is specified in the document; "PROPOSED MPEG DOS APPLICATION PROGRAMMING INTERFACE" by Creative Technology Ltd. .

The DOS Driver will largely consist of the same code modules as the Windows MCI Driver. Modifications will include the driver structure (a DOS TSR instead of a Windows Installable Driver), memory management, interrupt installation and some command and command handling differences.

The DOS MPEG Player performs similar functions to the Windows Media Player program. It will operate in a standard VGA graphic mode and provide the following controls:
File Open, Play, Pause/Resume, Stop, Fast Fwd, Rewind.

5) **CD-ROM Characterization**
It is important that the software play just as well on all CD-ROMs (SONY, Mitsumi etc.). We have found in our experience that various CD-ROMs display different characteristics for reading Video CDs. e.g., Mitsumi has on board cache of 16K while SONY has on board cache of 64K. Manufacturers may have to decide on the range of CD-ROMs and PC systems that this particular software may have to be qualified.

4.0 Conclusion

The explosion of market for interactive media products will hasten the migration of MPEG decoder hardware towards PCI interface. The higher performance buses have been able to provide the extra throughput needed to bring real motion to many video applications. Unfortunately the bottleneck in the proliferation of the MPEG technology in the consumer arena is severally limited by the lack of software in the industry. Software is once more a critical element in determining the extent of adoption by the industry.

5.0 References

1. PCI Specification 2.0, April 30 1994, Published by the PCI Consortium
2. "Hollywood Honchos think DVD's a star",EE Times, January 30, 1995, Issue 833
3. "PCI Bus Speeds Peripheral Communications", John Gallant, EDN December 8, 1994, pp. 93-98
4. "Treat Circuit Boards as Design Components in PCI-based Systems", Jim Murashige, EDN November 23, 1994, pp. 111-116.
5. " Software/Hardware Methodology for MPEG Decoders for Windows based systesm", Margi Systems Inc.'s internal design document.

PCI IMPEG IMPLEMENTATIONS

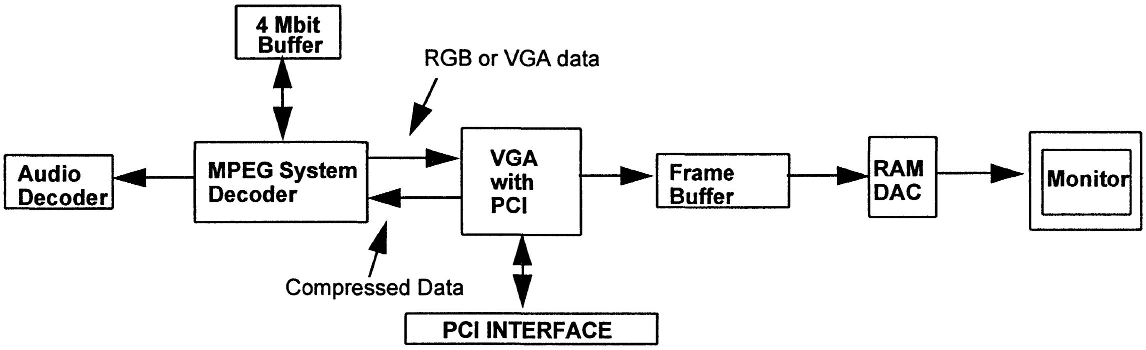


Figure 2. Integrated MPEG Accelerated VGA

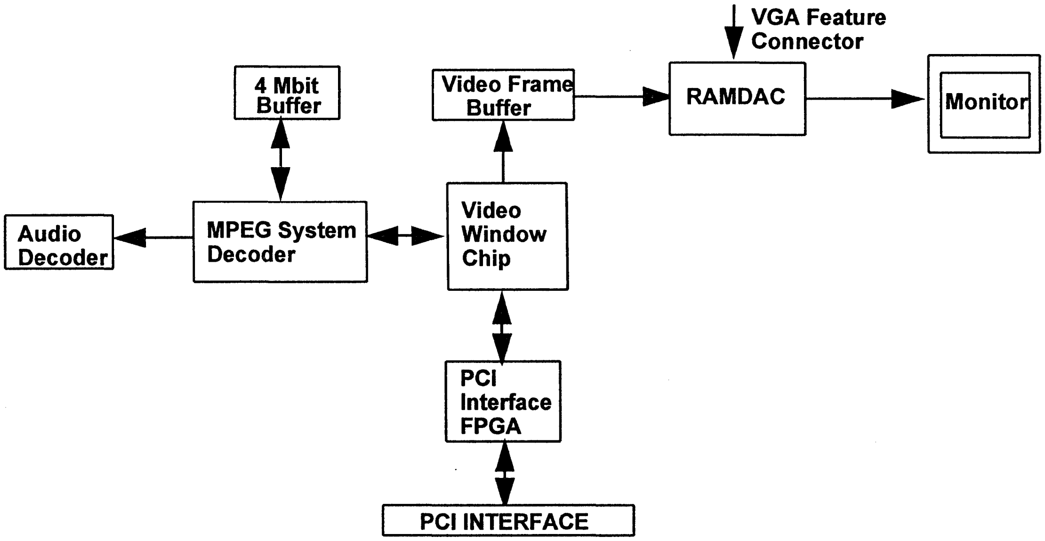


Figure 3. MPEG Video in a Window System

Distributed DMA in PCI systems
[and Legacy IRQ support]

David Evoy
Manager of Architecture
VLSI Technology
8375 South River Parkway
Tempe, AZ 85284
Ph. (602) 752-6241 Fax (602) 752-6002

Mr. Evoy is the Manager of Architecture for VLSI Technology's Personal Computer Division. He has been with VLSI for four years. For the 14 years prior to joining VLSI, Mr. Evoy was a founder and Vice President of Engineering for Konan Corporation.

VLSI TECHNOLOGY, INC.

Distributed DMA

in PCI systems

[and Legacy IRQ support]

The DMA problem

- ❑ **PCI does not support legacy DMA**

- ❑ **Legacy DMA uses Fly - by data transfers**
  - MEMR# and IOWR# asserted with data

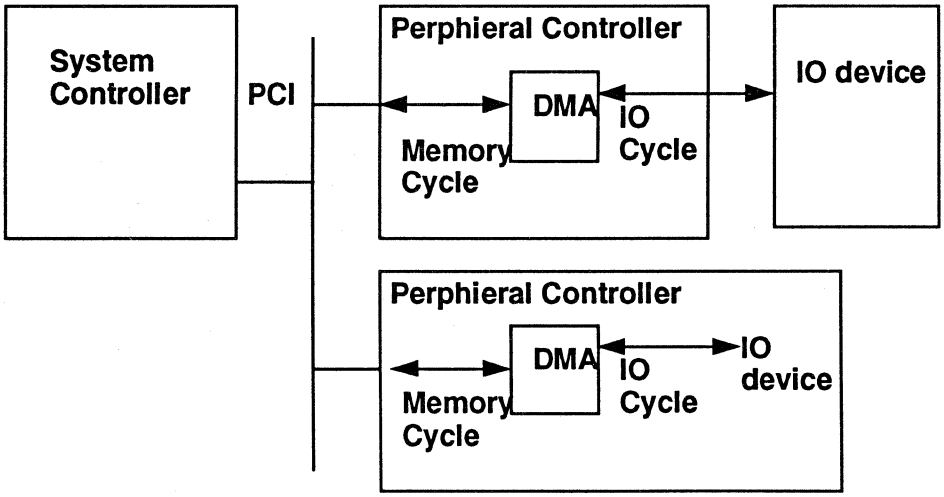
    or
  - MEMW# and IORD# asserted with data
  - PCI cannot support fly by transfers

- ❑ **PCI does not have DMA signals**
  - No DACK# or DRQ# signals on PCI
  - Not enough pins available on PCI connector

Who needs a solution?

- ❑ **PCMCIA PCI controllers**
  - Allows support of DMA devices
- ❑ **PCI docking systems**
  - Allows PCI docking connector without DRQ# and DACK# signals
- ❑ **Super IO chips**
  - DMA needed for floppy and printer support.
- ❑ **PCI audio cards / PCI audio mother board devices**
  - Can maintain compatibility with current DMA drivers
- ❑ **Systems without an ISA cage**
  - Allows full legacy compatibility without an ISA bus

What is distributed DMA?



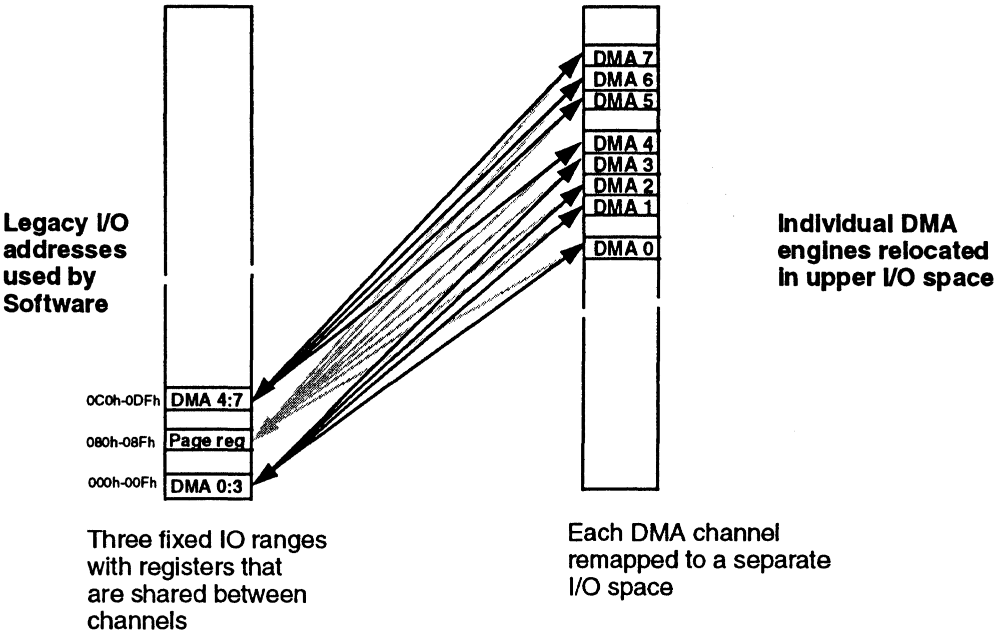
DMA channels are distributed into the peripheral chip(s).

Peripheral chip does PCI master cycles, IO portion of the DMA cycle is between Peripheral chip and external device or internal to peripheral chip.

DMA legacy programming model is maintained via centralized I/O mapper.

I/O MAPPING

The two 8237s are mapped into 8 separate IO address spaces



Legacy I/O addresses used by Software

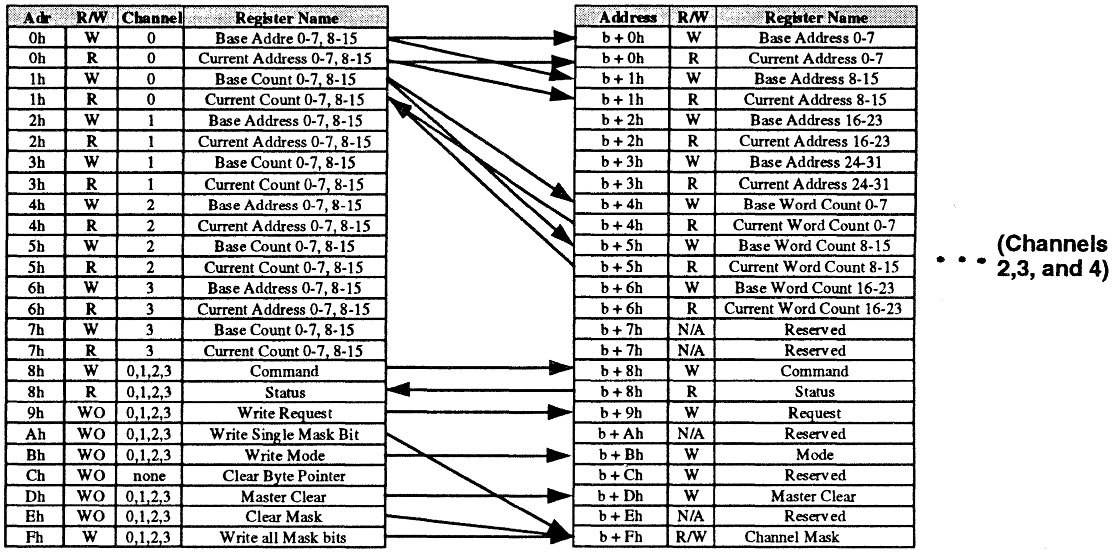
Individual DMA engines relocated in upper I/O space

Three fixed IO ranges with registers that are shared between channels

Each DMA channel remapped to a separate I/O space

Single 8237 (4 channels)

Distributed DMA "slice" (1 channel)



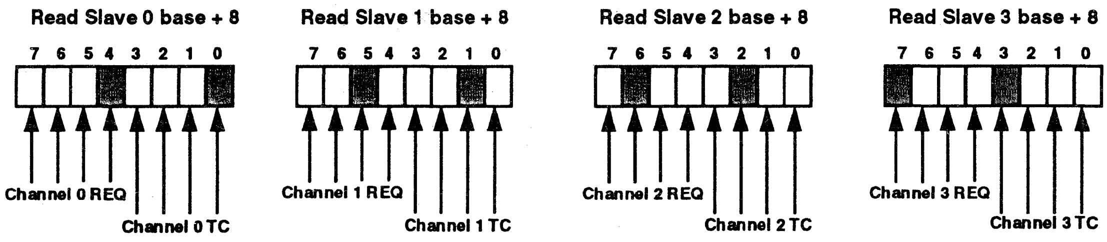
Adr	R/W	Channel	Register Name
0h	W	0	Base Addre 0-7, 8-15
0h	R	0	Current Address 0-7, 8-15
1h	W	0	Base Count 0-7, 8-15
1h	R	0	Current Count 0-7, 8-15
2h	W	1	Base Address 0-7, 8-15
2h	R	1	Current Address 0-7, 8-15
3h	W	1	Base Count 0-7, 8-15
3h	R	1	Current Count 0-7, 8-15
4h	W	2	Base Address 0-7, 8-15
4h	R	2	Current Address 0-7, 8-15
5h	W	2	Base Count 0-7, 8-15
5h	R	2	Current Count 0-7, 8-15
6h	W	3	Base Address 0-7, 8-15
6h	R	3	Current Address 0-7, 8-15
7h	W	3	Base Count 0-7, 8-15
7h	R	3	Current Count 0-7, 8-15
8h	W	0,1,2,3	Command
8h	R	0,1,2,3	Status
9h	WO	0,1,2,3	Write Request
Ah	WO	0,1,2,3	Write Single Mask Bit
Bh	WO	0,1,2,3	Write Mode
Ch	WO	none	Clear Byte Pointer
Dh	WO	0,1,2,3	Master Clear
Eh	WO	0,1,2,3	Clear Mask
Fh	W	0,1,2,3	Write all Mask bits

Address	R/W	Register Name
b + 0h	W	Base Address 0-7
b + 0h	R	Current Address 0-7
b + 1h	W	Base Address 8-15
b + 1h	R	Current Address 8-15
b + 2h	W	Base Address 16-23
b + 2h	R	Current Address 16-23
b + 3h	W	Base Address 24-31
b + 3h	R	Current Address 24-31
b + 4h	W	Base Word Count 0-7
b + 4h	R	Current Word Count 0-7
b + 5h	W	Base Word Count 8-15
b + 5h	R	Current Word Count 8-15
b + 6h	W	Base Word Count 16-23
b + 6h	R	Current Word Count 16-23
b + 7h	N/A	Reserved
b + 7h	N/A	Reserved
b + 8h	W	Command
b + 8h	R	Status
b + 9h	W	Request
b + Ah	N/A	Reserved
b + Bh	W	Mode
b + Ch	W	Reserved
b + Dh	W	Master Clear
b + Eh	N/A	Reserved
b + Fh	R/W	Channel Mask

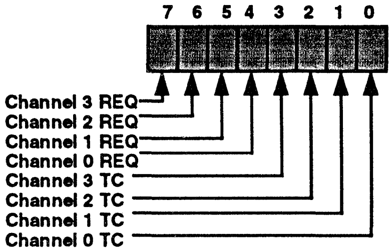
Example of a Read Status Command

Host reads DMA status at IO address 8

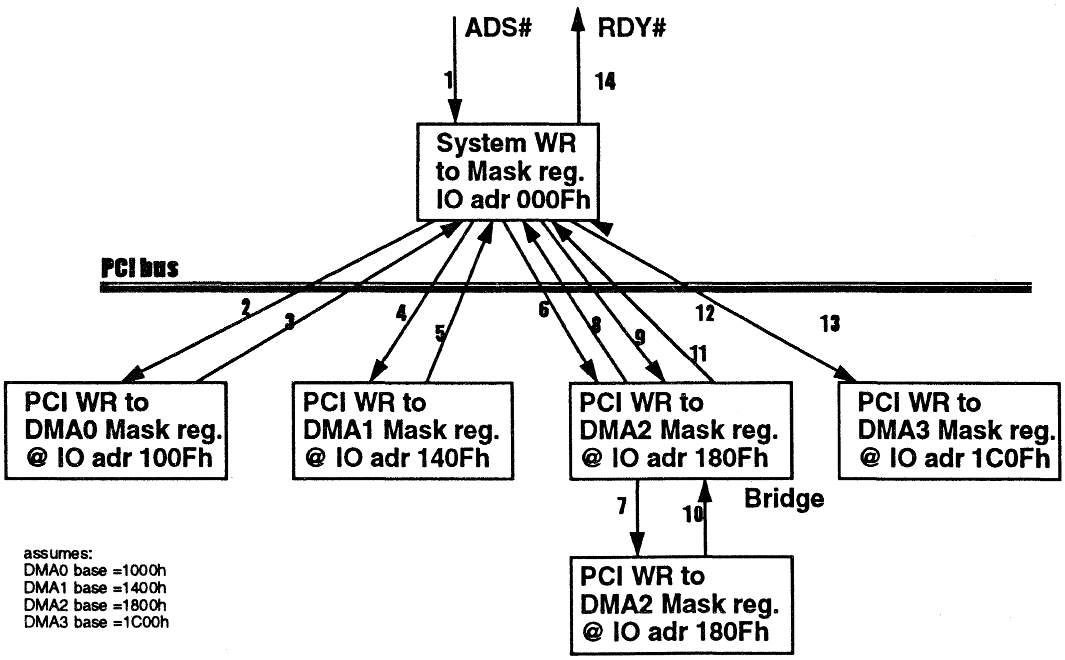
System controller breaks this into 4 separate IO reads



System controller assembles four reads into a single byte

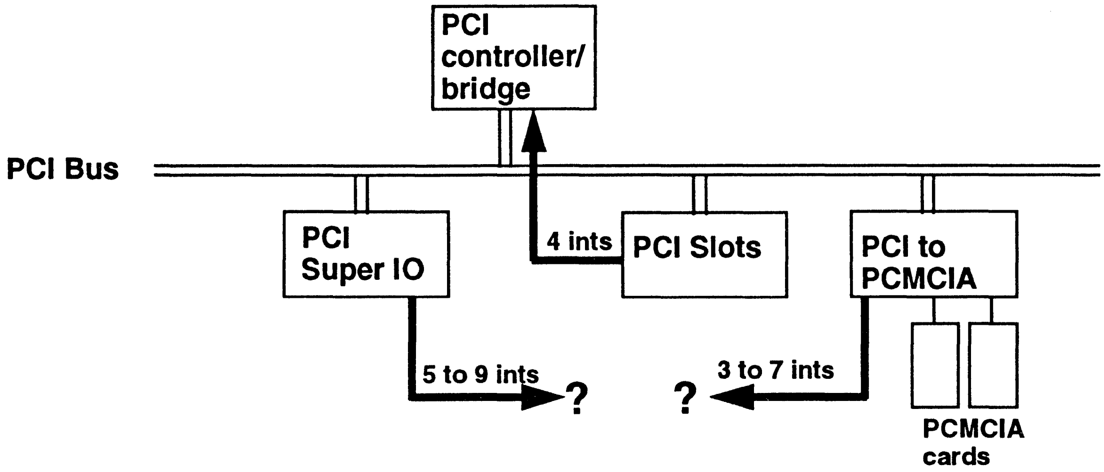


Write to a register that is shared by four DMA channels


ADS#
RDY#
1
14
System WR to Mask reg. IO adr 000Fh
PCI bus
2
3
4
5
6
8
9
12
11
13
PCI WR to DMA0 Mask reg. @ IO adr 100Fh
PCI WR to DMA1 Mask reg. @ IO adr 140Fh
PCI WR to DMA2 Mask reg. @ IO adr 180Fh
PCI WR to DMA3 Mask reg. @ IO adr 1C0Fh
7
10
Bridge
PCI WR to DMA2 Mask reg. @ IO adr 180Fh

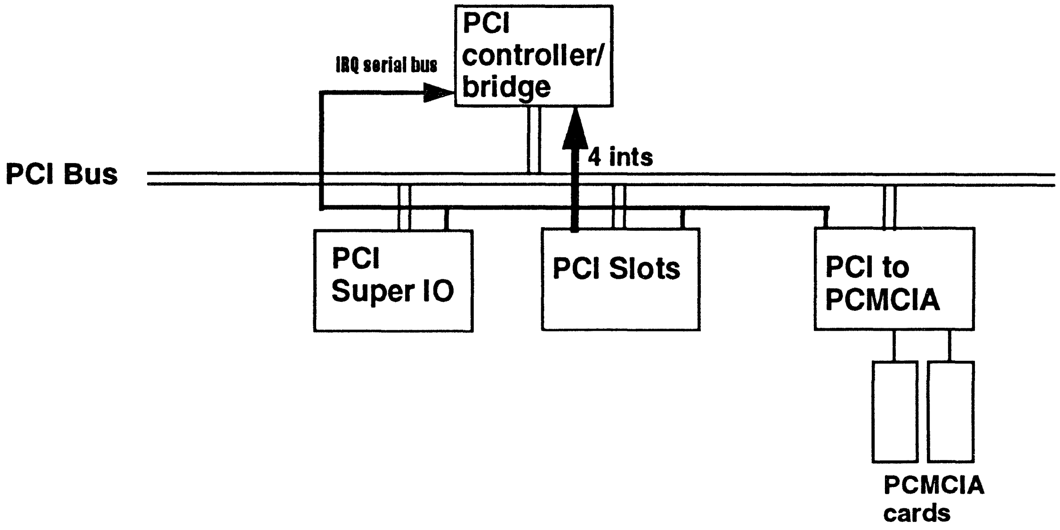
assumes:
DMA0 base =1000h
DMA1 base =1400h
DMA2 base =1800h
DMA3 base =1C00h

The IRQ Problem



- ❑ PCI has 4 sharable IRQs
- ❑ ISA bus has 11 edge sensitive IRQs
- ❑ ISA style IRQs are required for PCMCIA, Audio, and Legacy devices
- ❑ Transitioning devices from ISA to PCI is constrained by IRQs
- ❑ Evolution of PCI / ISA systems to PCI only possible with IRQ expansion

The serialized IRQ solution for PCI



A single pin serial IRQ bus interconnects all devices.
Saves pins
Increases Plug-n-Play support
Docks using a smaller connector
Allows true legacy IRQ support in the PCI bus

PCI legacy DMA and IRQs, one or two issues?

- □ **DMA and IRQ solutions appear independent - - but - -**
- □ **Docking**
  - ◆ PCI does not provide DRQs, DACKs, or adequate IRQs so numerous side band signals are required, making PCI docking overly expensive.
- □ **PCI card slot**
  - ◆ PCI card slots do not provide DRQ, DACK#s, or IRQs
- □ **Super IO chips for PCI bus**
  - ◆ Super IO chips need IRQ and DMA support to complete the transition to the PCI bus
- □ **PCMCIA**
  - ◆ Need to address both IRQs and DRQs to solve PCMCIA interface problems
- □ **Conclusion - Coordinated solutions for both problems are required.**

Summary

- □ **Distributed DMA**
  - ◆ Improves PCI bandwidth
  - ◆ Extensible, including 32 bit extensions, bursting
  - ◆ One PCI cycle for 1 to 4 DMA cycles
  - ◆ Strong industry support
  - ◆ Enables moving Floppy, Audio and PCMCIA to PCI bus
- □ **Serialized IRQ for PCI systems**
  - ◆ Decreases number of pins required to support Plug-n-Play on multiple devices
  - ◆ Enables placing PC AT legacy devices on the PCI bus
  - ◆ Will enable placing PC AT legacy devices on PCI cards when/if a pin is allocated to IRQSER

SINGLE-CHIP, GENERAL-PURPOSE INTERFACE TO THE PCI BUS

John Williams
AMCC
6195 Lusk Blvd.
San Diego, CA 92122

ABSTRACT

This paper presents an overview of the benefits of the PCI bus to new, high-performance PC applications. The discussion focuses on meeting the requirements of the PCI specification and the advantages or disadvantages of various approaches used to interface to the PCI bus. Finally, a general-purpose, single-chip solution for interfacing to the PCI bus is presented, with various modes of operation and add-on card interface options detailed.

INTRODUCTION

In the past, different system architectures contained different bus standards. Intel architecture systems used the ISA, EISA, or Multibus. Motorola systems used the VME bus, and numerous other architectures each implemented their own standard. Add-in card manufacturers were forced to make a different product for each system architecture they wanted to plug into. The PCI (Peripheral Component Interconnect) bus standard is being adopted for use in Intel architecture, Power PC, DEC Alpha and other systems, allowing a single card design to be used in multiple system architectures.

A limitation of previous bus standards was performance. The bandwidth provided by previous PC bus standards was inadequate for many new application areas such as ATM, RAID and multimedia. Newer bus standards, such as the VESA VL bus allow much greater bandwidth, but still have limitations.

The PCI bus allows add-in card manufacturers to create designs which can be used in multiple system architectures, providing a larger market for add-in card vendors. PCI also has the bandwidth required for many new, high-performance applications and provides for growth to 64-bits, 66 MHz, and 3.3V operation to support future performance requirements.

PCI COMPLIANCE

To ensure that PCI devices (either on the mother-board or on add-on cards) are useable in multiple system architectures, they must meet an extensive set of guidelines. The PCI Specification defines what is required to interface to the PCI bus. Some of the major areas involve:

Signal definitions
Bus Commands and transactions
Bus arbitration
Cache support
Configuration space definition
Electrical interface

The current specification is reasonably specific, but like all great literary works, there is room for interpretation. As the PCI specification evolves, it will become more clearly defined, but ambiguities introduce difficulties for add-in card manufacturers interfacing to the PCI bus. If a BIOS vendor interprets the specification one way and a hardware vendor interprets it another, problems occur.

To help avoid the problems with different interpretations of the PCI specification, the PCI SIG (Special Interest Group) publishes an extensive "PCI Compliance Checklist." The Compliance Checklist lists requirements for PCI Compliance. PC, BIOS, add-on card, and silicon vendors participate in PCI compliance workshops to ensure they meet PCI interface requirements and function together in a system environment. This allows differences in interpretations to be identified and worked-out prior to production.

CONNECTING TO THE PCI BUS

Creating an interface to the PCI bus is not a trivial task. There are a number of standard devices available to translate PCI to SCSI, PCI to PCMCIA, etc., but there is a conspicuous absence of general-purpose solutions. What about the add-on board manufacturer with an 8051, 68000, 80960, a PC peripheral device, or a DSP on their card? A general solution is required to interface these devices to the PCI bus. There are three possibilities for this interface: ASIC, FPGA, and off-the-shelf devices.

ASIC Solution

The designer could create an ASIC which provides the logic required to interface to the PCI bus. This approach has the advantage of allowing all glue logic required for the add-on to be implemented in a single device, but there is a significant drawback to this approach: cost.

The primary cost component of an ASIC approach is the NRE. The size of an NRE is prohibitive to all but very high volume designs. A secondary cost involves development time. The significant amount of time required to develop and debug a PCI compliant device can lead to increased time-to-market, resulting in lost revenue and reduced marketshare. There are significant risk factors to an ASIC implementation.

FPGA Solution

Another possibility is to create the PCI bus interface with field programmable gate arrays. This approach also allows the designer to customize the add-on interface for their application, but, again, has the disadvantage of cost.

As with an ASIC solution, a major component of the cost for an FPGA solution is development time. The PCI specification must be converted into logic and then debugged. Some FPGA manufacturers provide the PCI interface logic block, but the designer is still required to implement and debug the logic needed to interface the add-on logic to the predefined PCI interface block.

Another significant cost of an FPGA solution is component cost. Currently FPGA densities are not high enough to support a full master/slave PCI interface on a single device. This leaves the designer with the option of using a defeatured PCI interface or using multiple FPGAs. If multiple FPGA devices are used, the issue of PCI electrical compliance arises. The PCI specification states that no PCI signal trace (on the add-on board) may be longer than 1.5 inches. Even with a single interface component this can be difficult to achieve and using multiple FPGA devices may require adding signal layers to the add-on board.

Like an ASIC solution, the development time for the PCI interface can increase time to market, leading to lost revenue and decreased market-share. Additionally, the cost of FPGA devices and their implementation increases the cost of an add-on board, significantly. This is a big disadvantage in a competitive market.

Off-the-Shelf Solution

The ideal interface to the PCI bus is an off-the-shelf, low-cost, standard product which is guaranteed to be PCI compliant. The AMCC S593x "PCI Matchmaker" devices fulfill these requirements. As AMCC is a member of the PCI SIG, the S593x products are tested in the "compliance workshops" with PCI systems, chipsets and BIOS'. This removes the burden of compliance testing from the add-on board manufacturer and greatly reduces development time by allowing the designer to focus on the actual application, not debugging the PCI interface.

The AMCC S593x devices provide a general-purpose interface to the PCI bus. The add-on bus interface is flexible enough to allow it to interface with 8-, 16-, or 32-bit add-on microprocessors or peripherals, and even allows operation without an add-on processor. The S593x can function as a PCI initiator or target, or both, and is capable of transferring data at the full PCI bandwidth.

The S593x provides three ways to communicate with the add-on card. There are four 32-bit mailbox registers, four configurable 8-, 16-, or 32-bit pass-thru regions and an eight-deep 32-bit FIFO (which also can be used in bus mastering applications). The device is configured at reset based on the contents of an external nvRAM (x8 or serial), which can also contain an expansion BIOS. The modes of operation are described in more detail later.

The S593x is a single-chip solution. Various versions are available with different add-on bus widths and configuration options.

The packages range from a 120-pin PQFP for the 16-bit add-on bus with a serial nvRAM interface (S5930) to a 160-pin PQFP for a 32-bit add-on bus with a x8 nvRAM interface (S5933). All devices support a 32-bit, 33 MHz PCI interface.

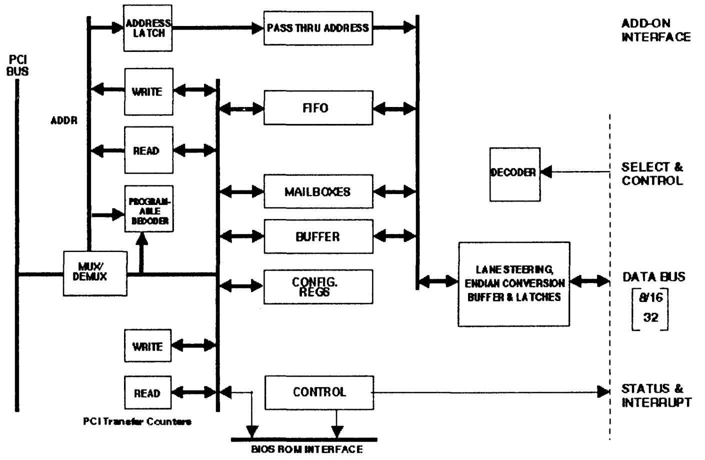


Figure 1-1. S593x Block Diagram

AMCC S593x ARCHITECTURE

The S593x provides a 32-bit PCI-compliant interface with a flexible 8-, 16-, or 32-bit multiplexed address/data bus on the add-on card side. A block diagram of the S593x architecture is shown in Figure 1. The following sections provide an overview of the S593x architecture and its modes of operation.

Register Architecture

The S593x contains three groups of registers: PCI Configuration Registers, PCI Operation Registers, and Add-on Operation Registers. These define the different modes of operation for the device.

PCI Configuration Registers. All PCI devices contain a 256 byte region called the Configuration Space. Some of the first 64 bytes (the header) are required for PCI compliance. The remaining 192 bytes are not defined and are device-specific. The 64 byte header contains the device vendor ID, device ID, revision number, and other information such as specific device capabilities and amount of memory required for operation. The configuration registers are intended for use during system initialization and catastrophic error handling.

PCI Operation Registers. The S593x contains 16 double word registers defined as PCI Operation Registers. These registers provide the primary communication path from the PCI bus to the add-on bus. Through these registers, the host CPU can access mailbox registers and the internal FIFOs. These registers also configure the S593x for bus mastering, define other operation and indicate status.

Add-on Operation Registers. The S593x contains 18 double word registers defined as Add-on Operation Registers. These registers provide the primary communication path from the add-on bus to the PCI bus. Through these registers, the add-on can access mailbox registers, the internal FIFOs, and address or data associated with pass-thru cycles. These registers also configure the S593x for bus mastering, define other operation and indicate status.

Non-Volatile Memory Interface

The S593x can be used in it's default configuration, or a custom configuration can be downloaded from a non-volatile memory at reset. The device can download configuration information from either a serial (2 Kbytes, maximum) or a byte-wide device (64 Kbytes, maximum). The presence of an external memory and its type (serial or byte-wide) is automatically detected by the 593x at reset, and configuration information is downloaded.

This interface allows add-on card vendors to program their own Vendor ID and Device ID information into the S593x (or AMCC's default values may be used). Other information which is read by the host during initialization is also downloaded, defining the characteristics of the add-on card. The non-volatile memory interface also provides for an optional expansion BIOS on the PCI bus.

As many non-volatile devices may be written, in-circuit, the S593x allows the non-volatile memory to be written from the PCI bus or the add-on bus. This provides the add-on card manufacturer the ability to perform field upgrades to the expansion BIOS code. This also allows the S593x configuration information to be tailored to a specific system architecture via software during installation.

Mailbox Operation

The S593x contains eight 32-bit mailbox registers. Four mailboxes transfer data from the PCI bus to the add-on bus, and four mailboxes transfer data from the add-on bus to the PCI bus. The mailbox registers are useful for transferring data, command or status information between the PCI and add-on buses.

Mailbox status (empty/full) may be monitored with status bits in the Add-on and PCI Operation Registers, or interrupts can be generated by mailbox accesses. A PCI interrupt (INTA#) can be generated by either an outgoing mailbox being read by the add-on or by an incoming mailbox being written by the add-on. An add-on interrupt (IRQ#) can be generated by either an incoming mailbox being written by the PCI bus or an outgoing mailbox being read by the PCI bus. Each interrupt condition can be individually enabled or disabled.

The empty/full condition (interrupt condition) for mailboxes is configurable. A specific byte is identified to cause the empty/full flag to be set. This works well for systems with an 8-bit or 16-bit add-on interface. For example, a PCI to add-on mailbox can be configured to indicate the full condition (and optionally generate an interrupt to the add-on) when Byte 0 is written by the PCI bus. This allows a simple interface to an 8-bit add-on bus, as data can be transferred in single-byte quantities without assembling or disassembling 32-bit data.

Some designs require a method to generate interrupts to the PCI bus without writing to a mailbox register (cards without a CPU on the add-on). The S593x may be configured to allow interrupts to be generated through hardware. Add-on outgoing mailbox 4, byte 3 is accessible via multiplexed function device pins. All 8 data bits and a load clock are provided. This allows the add-on to toggle the load clock and generate an interrupt to the PCI bus (provided the mailbox full condition is set for mailbox 4, byte 3).

Pass-thru Operation

Pass-thru operation allows PCI bus cycles to be executed in real time with the add-on interface. The PCI bus may directly read from or write to add-on resources. The S593x allows definition of up to four individual pass-thru regions in PCI memory or I/O space. Each region may be defined as 8-bits, 16-bits, or 32-bits wide and mapped into memory or I/O space.

Pass-thru mode is a PCI target-only mode, making it useful for converting existing ISA/EISA designs to PCI (which, in many cases, do not act as bus masters). Pass-thru mode uses handshaking to communicate with the add-on interface. The pass-thru signals indicate when the PCI is writing to the add-on or needs to read from the add-on. They also provide information such as which specific byte lanes are involved in the transfer, if the access is a read or write, which of the four pass-thru regions is being accessed, and whether the access is a burst or a single cycle. Address and data information is passed between the PCI bus and add-on via add-on operation registers.

The add-on interface provides a pass-thru attention signal which indicates that the PCI bus is attempting to access an add-on resource. Once a pass-thru access is identified, the add-on interface can read the PCI address associated with the cycle (via the pass-thru address register) and read or write the appropriate data (via the pass-thru data register). Once the access is complete (all bytes read or written, in the case of a non-32-bit add-on), the add-on returns a pass-thru ready indication, ending the current access.

Pass-thru accesses can be either single data phase PCI cycles or PCI bursts. Provided the add-on logic can process data quickly enough, the pass-thru interface can support the full PCI bandwidth.

FIFO Operation

FIFO operation provides two FIFO data paths between the PCI and add-on buses. One FIFO passes information from the PCI bus to the add-on bus, and the other FIFO passes information from the add-on bus to the PCI bus. Each FIFO is 32-bits wide and 8 Dwords deep. The FIFOs may operate as either a PCI target or a PCI initiator (bus master).

Endian Conversion. The FIFO interface also allows endian conversion. The FIFO can be programmed for 16-bit, 32-bit, or 64-bit endian conversion on incoming and outgoing data. This allows an add-on processor and the host to operate in their native endian format.

16-bit and 8-bit Add-ons. The FIFO also interfaces easily to 16-bit add-ons. If the add-on is configured as 16-bits, the FIFO internally steers data in the upper 16-bits of the Dword to the lower 16-bits on alternate accesses. Another method to interface to 8-bit or 16-bit add-ons is available. The advance condition for the FIFO is programmable. The FIFO pointer is updated when a specific byte is accessed. By changing the advance condition to byte 0 or byte 1, the FIFO can be implemented as 8-bits or 16-bits wide.

FIFO Interface. To allow high transfer rates through the FIFO and a simple interface to an external FIFO, two signals are implemented. The RDFIFO# and WRFIFO# inputs provide a direct method to access the FIFO. The S593x can be programmed to allow either an asynchronous or synchronous FIFO interface with these signals. The FIFOs can also be accessed like the other add-on operation registers using the chip enable, address inputs, and read or write strobes.

Bus Mastering. The FIFOs allow bus mastering on the PCI bus. The PCI to add-on FIFO and the add-on to PCI FIFO each have an associated address register and transfer count register. These are initialized before each transfer. These registers may be loaded from either the host CPU or the add-on. This is determined at reset. To determine when a transfer is complete, the transfer count status may be monitored via a status bit, or an interrupt may be generated (the interrupt will be to the add-on or PCI bus, based on which side controls the address and transfer count loading).

The FIFO management scheme determines how full or empty the FIFO must be before it asserts REQ# to the PCI bus (to gain control of the bus). The PCI to add-on FIFO can be configured to request the bus when any of the 8 Dwords are empty, or only when four or more Dwords are empty. The add-on to PCI FIFO can be configured to request the bus when there is data in any of the 8 Dwords, or only when four or more Dwords are full. This allows the designer to control how often the S5933 requests the bus. The S5933 always attempts to perform burst operations to empty or fill the FIFOs.

Summary

Because the PCI bus applies to numerous system architectures, it allows a single add-in card hardware design to be created for multiple platforms. PCI also provides the bandwidth required for many new, high-performance applications.

Developing and debugging a compliant interface to the PCI bus is not a small task.

Depending on time constraints, cost constraints, and application requirements, different approaches are available. The competitive add-in card market requires low cost, and short design cycles. This makes an off-the-shelf PCI interface solution very attractive.

The AMCC S593x products provide a flexible, low-cost, compliant interface to the PCI bus. The architecture of the S593x makes it an excellent choice for cards being converted from the ISA/EISA standard, as well as newer applications requiring high data rates and bus mastering capabilities. This solution allows the hardware developer to focus on the actual application development rather than debugging the PCI bus interface logic. This significantly shortens design cycles and decreases development costs.

Biography

John Williams has a BSEE from Vanderbilt University. He worked for 4 years at Intel Corporation as an applications engineer in the Embedded Processor Division. Experience includes a number of 80C186Ex and 80386EX board designs as well as work on 3.3 Volts devices and the JEDEC JC-16 committee (low voltage). Currently he is working at Applied Micro Circuits Corporation (AMCC) as an applications engineer in the Computer Products Division, focusing on PCI devices.

THE APOLLO PLUS CHIP SET

Wen-Chi Chen
VIA Technologies, Inc.
5020 Brandin Court
Fremont, CA 94538 USA

ABSTRACT

The Apollo Plus chip set is a cost-effective implementation of a high performance and energy efficient PCI/ISA desktop personal computer system based on the 64-bit P54C/Pentium/K5/M1 super scalar processors. Either 3.3v or 5v CPU and cache interfaces are supported at external bus speeds up to 66MHz with internal CPU speeds up to 100MHz and above.

The four-piece chip set includes a system controller, a PCI bus controller (including integrated enhanced IDE and plug and play controllers) and two data buffers. The CPU bus is minimally loaded with only the CPU, secondary cache and the system logic chip set. The data buffers isolate the CPU bus from the DRAM, PCI and ISA buses so that CPU and cache operations will run reliably at the high frequency demanded by the processors.

Apollo Plus supports either single or dual processors based on write-back primary cache. Its key features are

- Integrated PCI buffer management
- Native and Concurrent IDE Controller
- Notebook class power management
- Emerging cache and DRAM support
- Integrated multimedia and native signal processing
- Plug and play ready
- Same chip set for multiplr platforms (Home/NB/MPC/DP/Server)

MEMORY CONTROL

Apollo Plus supports eight banks of DRAMs up to 512MB. The eight banks are grouped into four pairs with an arbitrary mixture of 256K/512K/1M/2M/4M/8M/16MxN DRAMs. Zero, one or both banks may be populated in each pair. The only constraint is that if both banks within the same pair are populated, they must be of the same type. This constraint fits particularly well with 72-pin x36 double side DRAM SIMM modules, although 30-pin x9 SIMM modules and 72-pin x36 single side SIMM modules are supported equally well.

DRAM data can be cached in the secondary cache based on on either synchronous or standard SRAMs from 128KB to 2MB. For synchronous SRAMs, 3-1-1-1 timing can be achieved for both read and write transactions up to 66MHz. For standard SRAMs, 3-2-2-2 and 4-2-2-2 timing can be achieved for interleaved read and write transactions up to 66mhz, respectively.

BUS INTERFACE

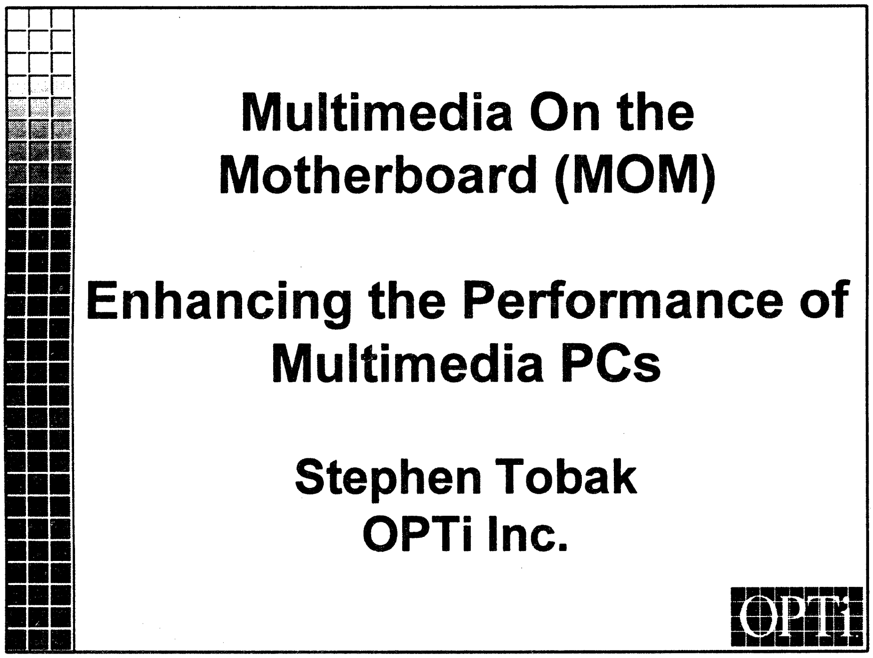
Apollo Plus supports a 32-bit PCI bus at a synchronous half the CPU frequency. The 64-bit to 32-bit data conversion and command regeneration are performed by the chip set's system controller with minimum overhead. Four levels of post write buffers are included between the CPU and the PCI bus to allow concurrent CPU and PCI operation. Consecutive CPU addresses are converted into burst PCI cycles with byte merging capability for optimal CPU to PCI throughput. A 32-bit fast data link is established between the two data buffers and the PCI bus controller so that the address, data and command information for CPU to PCI bus transctions is contained in the same chip. This arrangeent, unique to the Apollo Plus chip set, is crucial in achieving zero wait state buffer movement and in implementing sophisticated and upgradable buffer management schemes such as byte merging.

OTHER FEATURES

The integrated power management unit monitors I/O events, DMA and PCI master request signals to detect the status of system activity. It is ideal for high performance, high quality, high energy efficient and high integration desktop and notebook PCI/ISA computer systems. The chip set provides two plug and play ports for converting non plug and play devices into plug and play devices on the main board. The chip set also interfaces directly with the VT82C416MV integrated clock generator, real time clock and keyboard controller.

Multimedia On the Motherboard (MOM)

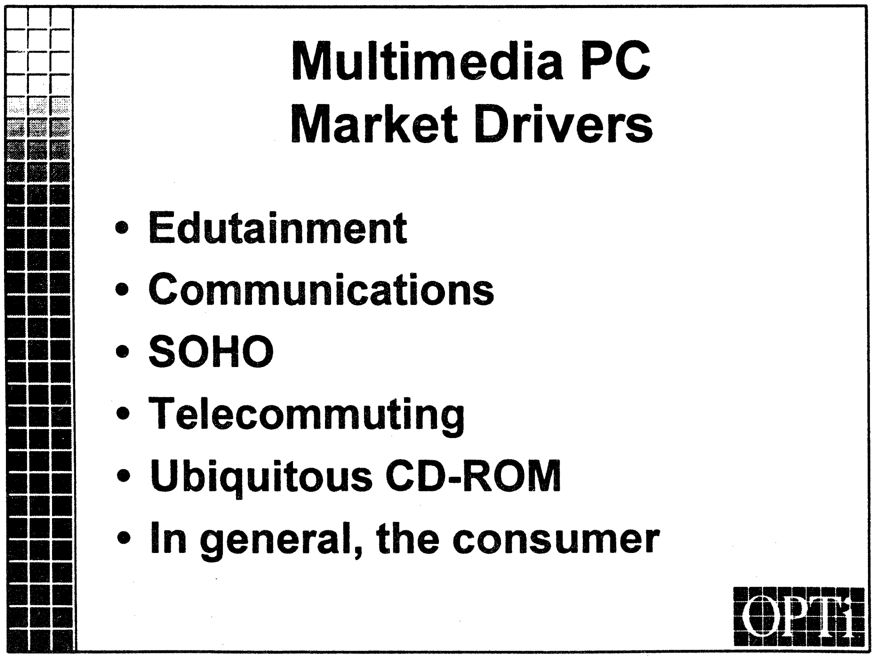
Stephen Tobak
Director of Corporate Marketing
OPTi Inc.
2525 Walsh Ave.
Santa Clara, CA 95051
408-486-8243



Multimedia On the Motherboard (MOM)

Enhancing the Performance of Multimedia PCs

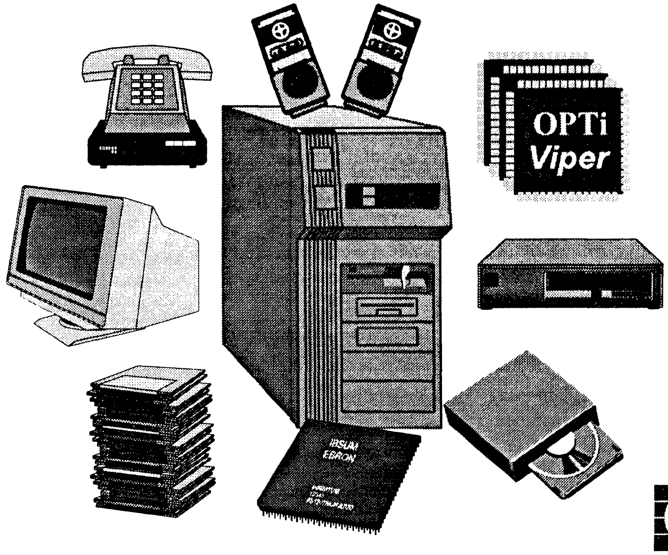
Stephen Tobak
OPTi Inc.



Multimedia PC Market Drivers

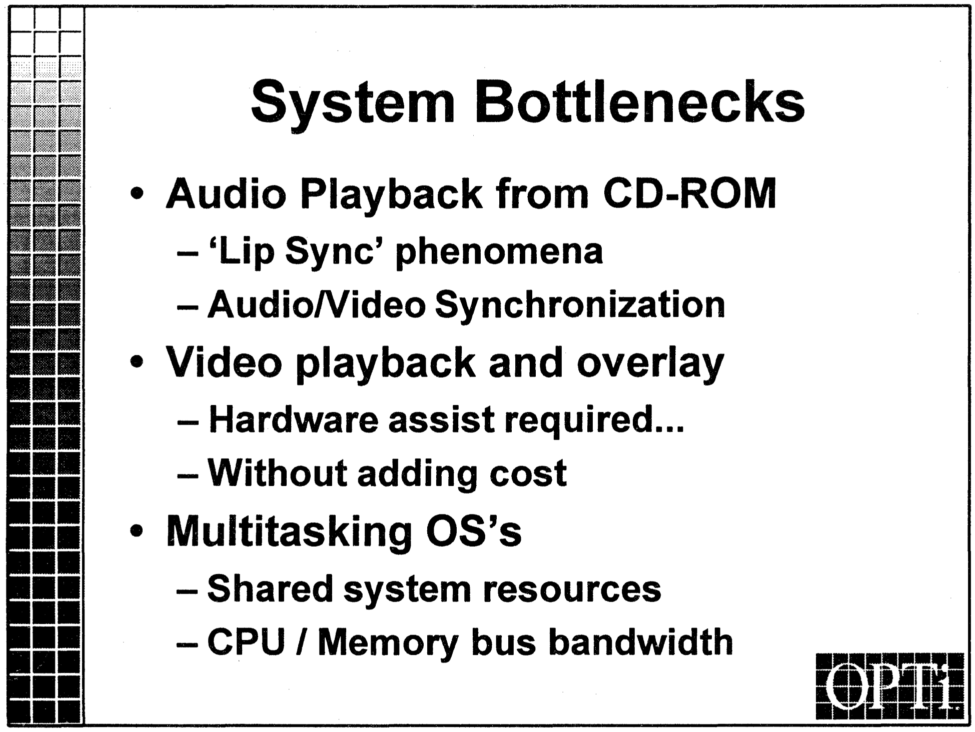
- Edutainment
- Communications
- SOHO
- Telecommuting
- Ubiquitous CD-ROM
- In general, the consumer

Multimedia System Components



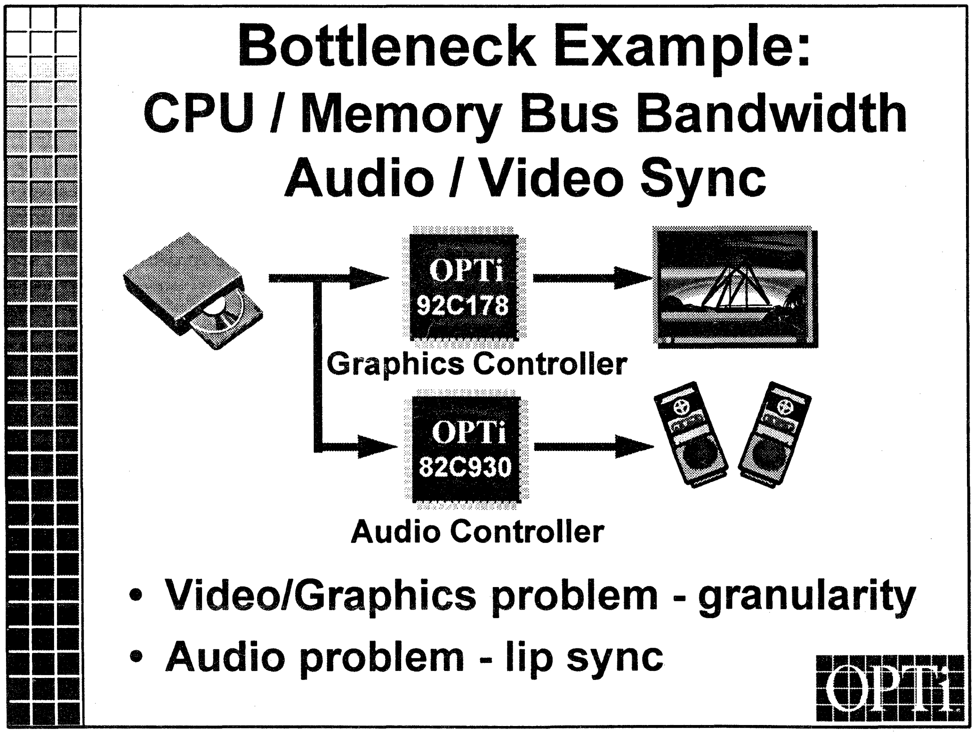
Do Today's Systems Provide Adequate Performance

- CPU/Memory performance
- Graphics performance
- Benchmarks emphasize single function/task performance, keeping other functions constant
- Performance metrics do not adequately measure multimedia systems performance



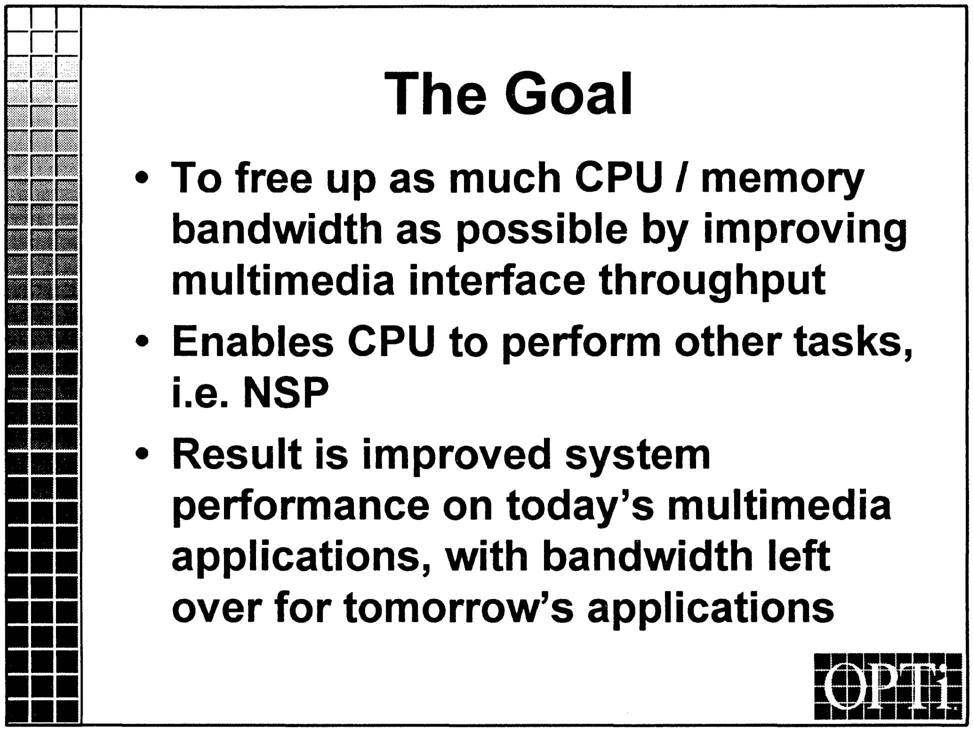
System Bottlenecks

- Audio Playback from CD-ROM
  - ‘Lip Sync’ phenomena
  - Audio/Video Synchronization
- Video playback and overlay
  - Hardware assist required...
  - Without adding cost
- Multitasking OS’s
  - Shared system resources
  - CPU / Memory bus bandwidth



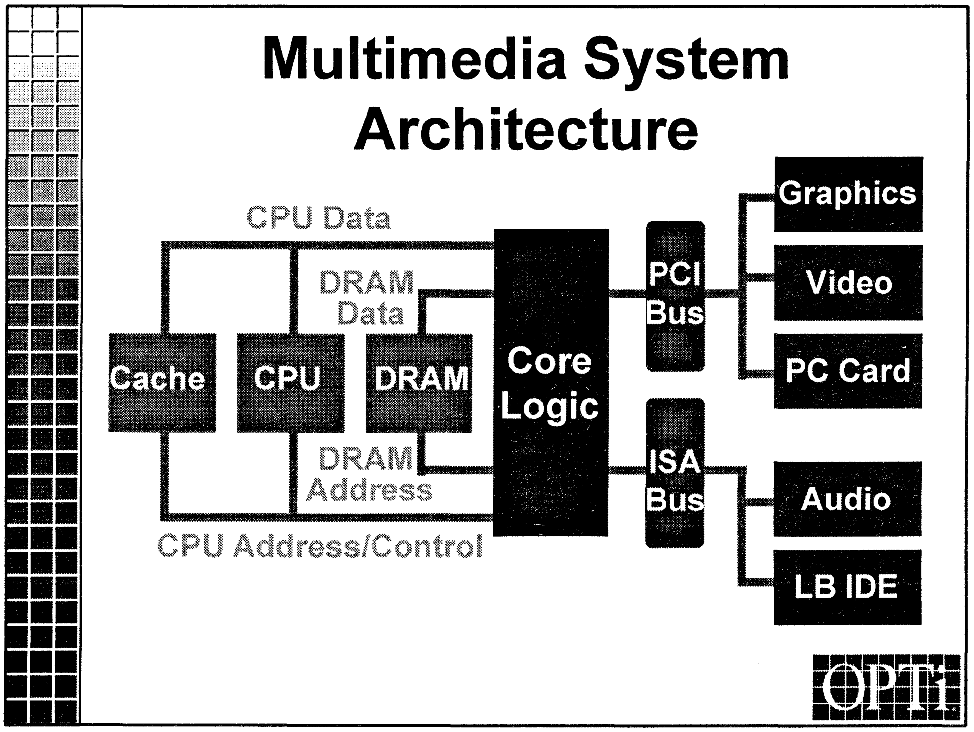
Bottleneck Example: CPU / Memory Bus Bandwidth Audio / Video Sync

- Video/Graphics problem - granularity
- Audio problem - lip sync



The Goal

- To free up as much CPU / memory bandwidth as possible by improving multimedia interface throughput
- Enables CPU to perform other tasks, i.e. NSP
- Result is improved system performance on today's multimedia applications, with bandwidth left over for tomorrow's applications



Multimedia System Architecture

CPU Data

DRAM Data

Cache

CPU

DRAM

Core Logic

DRAM Address

CPU Address/Control

PCI Bus

ISA Bus

Graphics

Video

PC Card

Audio

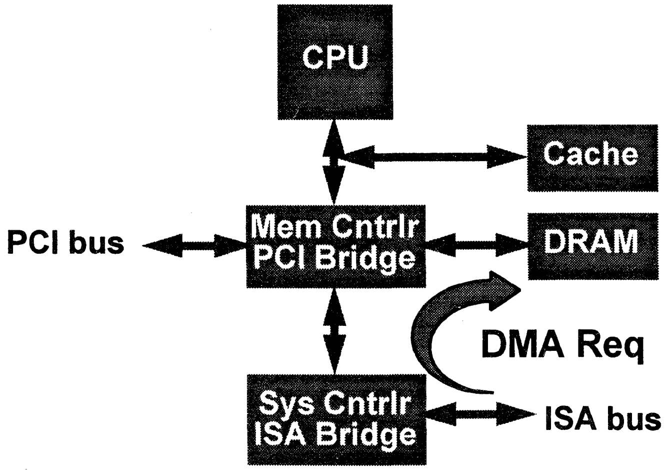
LB IDE

Key Audio Factors

- Real-time data
- 'Timeliness' on PC bus critical
  - Ear sensitive to audio aberations
  - Audio/video synchronization - 'lip sync'
- CD-ROM playback most difficult
  - Highest audio sampling rate
- DMA mechanism optimum

Audio Bottleneck

> 17.6% CPU / Memory bus bandwidth utilized in DMA operation



CPU

Cache

PCI bus

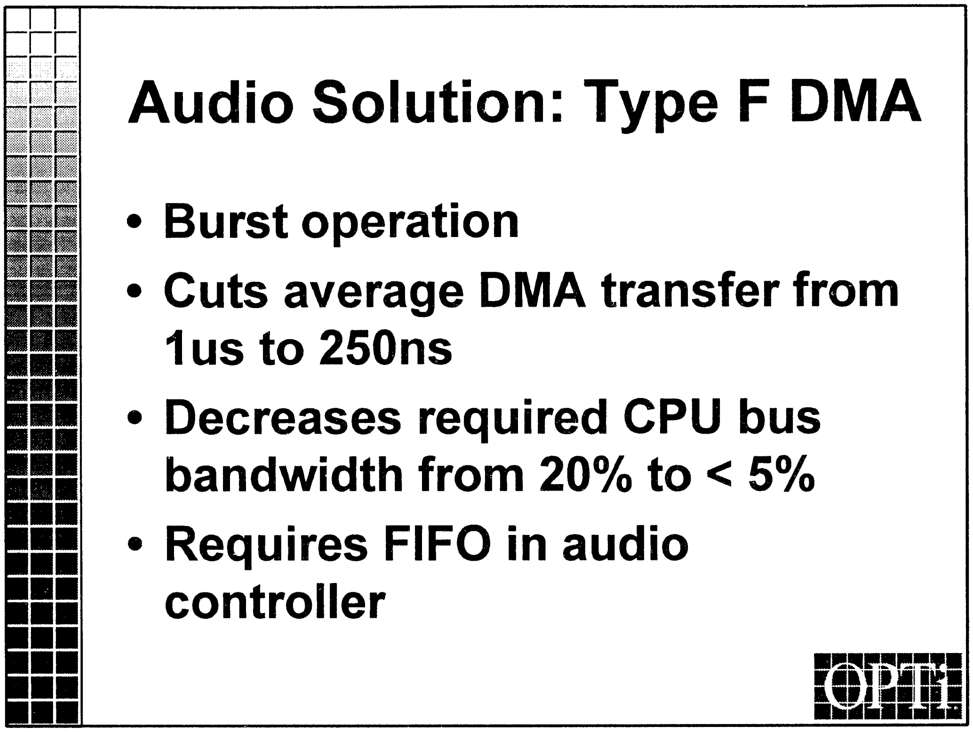
Mem Cntrlr
PCI Bridge

DRAM

DMA Req

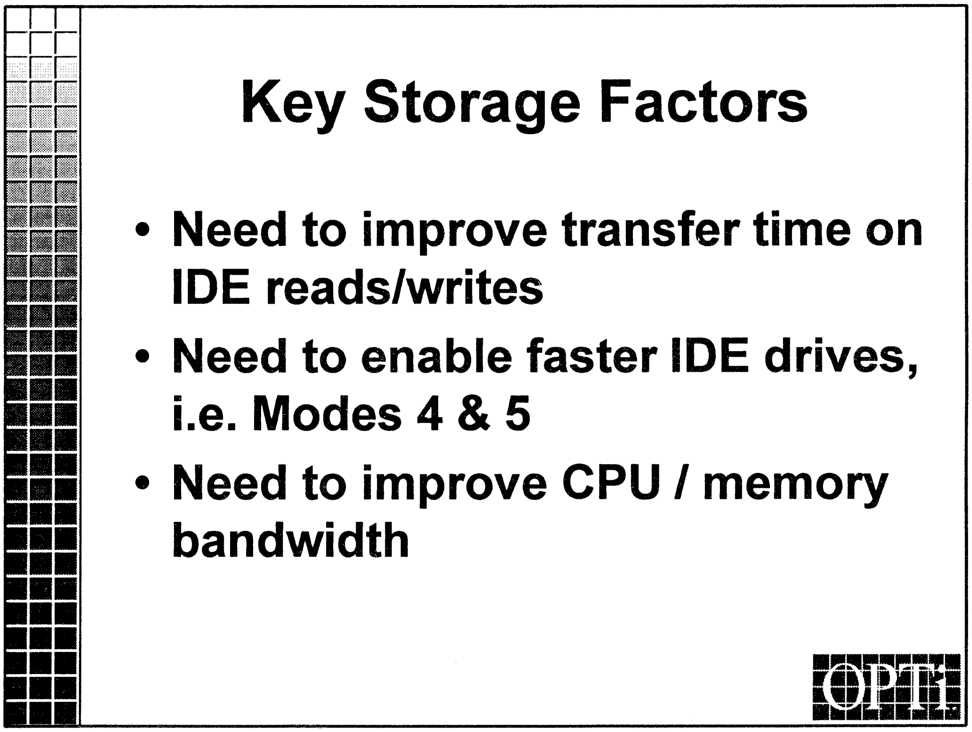
Sys Cntrlr
ISA Bridge

ISA bus



Audio Solution: Type F DMA

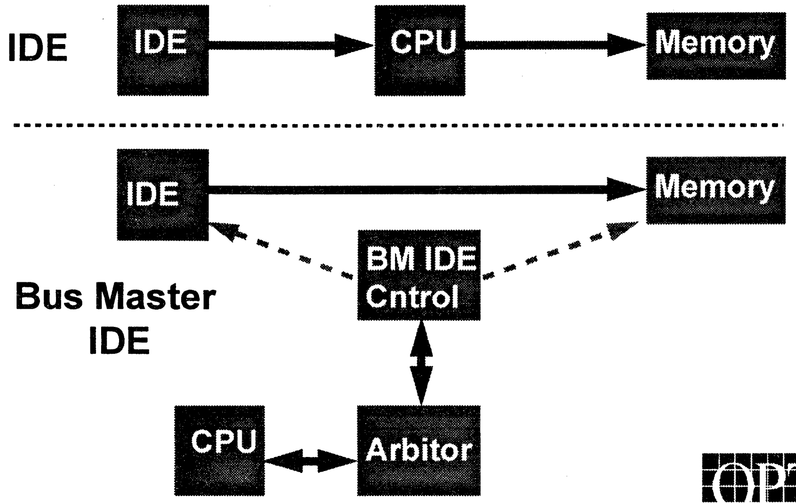
- Burst operation
- Cuts average DMA transfer from 1us to 250ns
- Decreases required CPU bus bandwidth from 20% to < 5%
- Requires FIFO in audio controller



Key Storage Factors

- Need to improve transfer time on IDE reads/writes
- Need to enable faster IDE drives, i.e. Modes 4 & 5
- Need to improve CPU / memory bandwidth

Storage Solution: Bus Master IDE



IDE

IDE → CPU → Memory

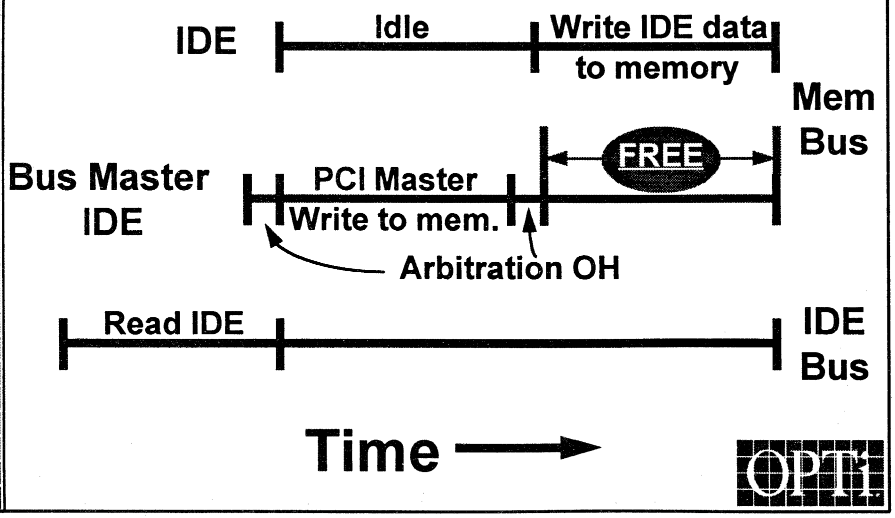
Bus Master IDE

IDE → Memory

BM IDE Cntrol

CPU ↔ Arbitor

Storage Solution: Bus Master IDE



IDE

Idle

Write IDE data to memory

Mem Bus

FREE

Bus Master IDE

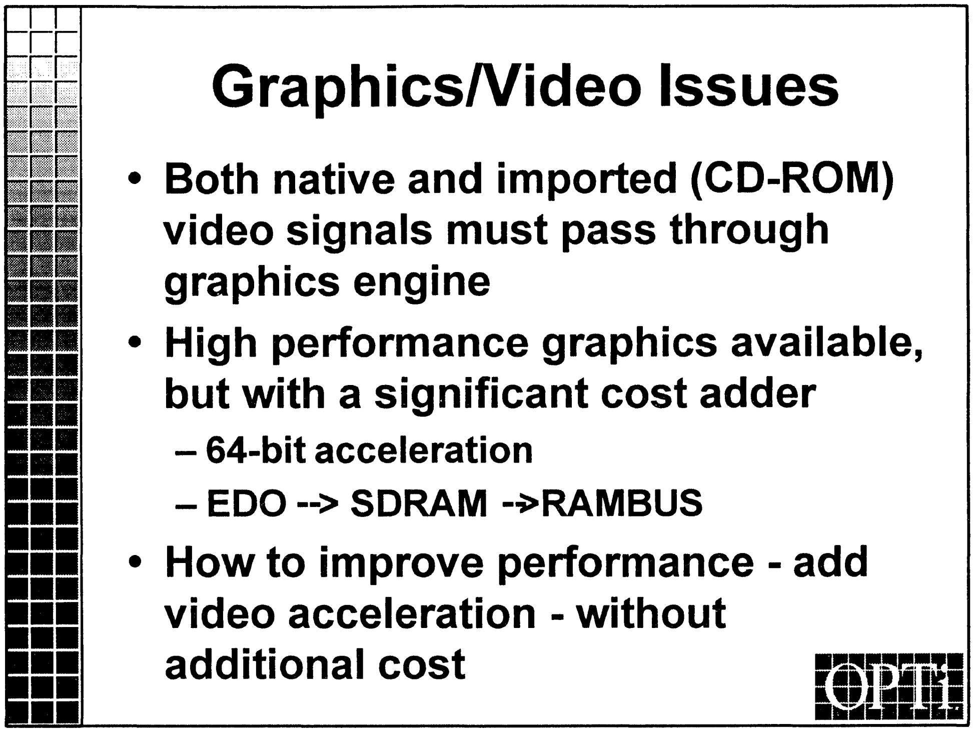
PCI Master Write to mem.

Arbitration OH

Read IDE

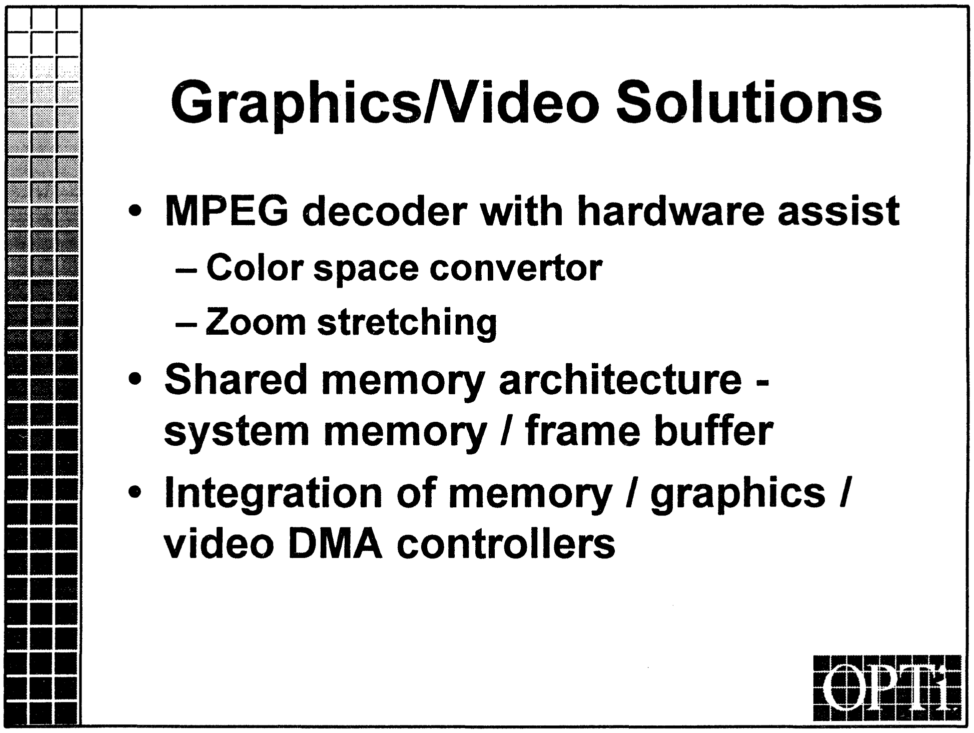
IDE Bus

Time →



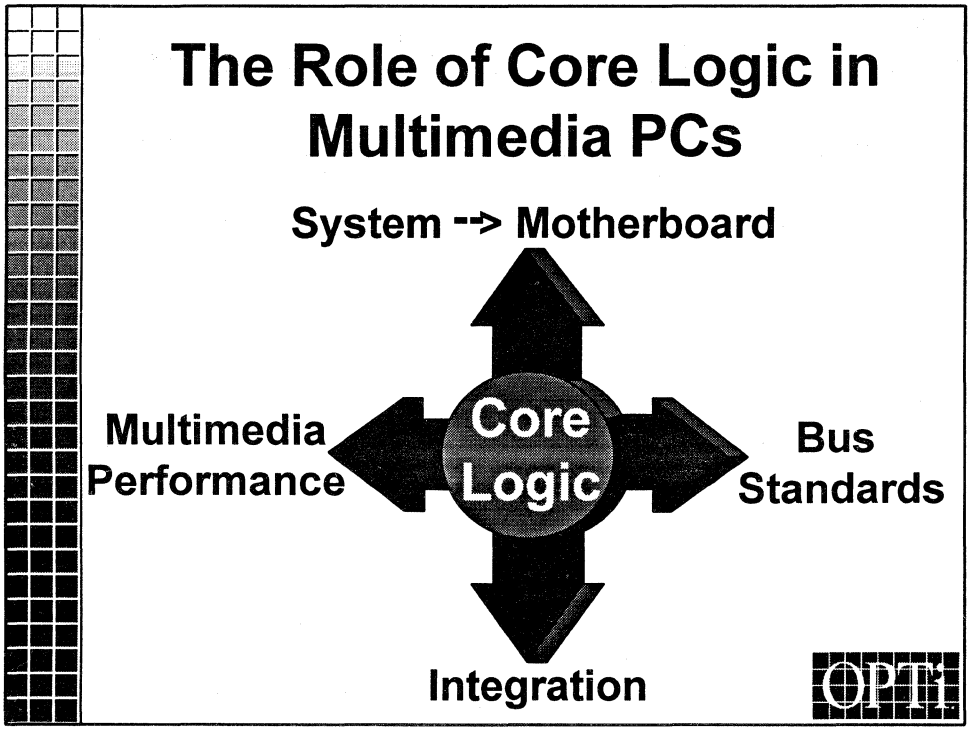
Graphics/Video Issues

- Both native and imported (CD-ROM) video signals must pass through graphics engine
- High performance graphics available, but with a significant cost adder
  - 64-bit acceleration
  - EDO --> SDRAM ->RAMBUS
- How to improve performance - add video acceleration - without additional cost



Graphics/Video Solutions

- MPEG decoder with hardware assist
  - Color space convertor
  - Zoom stretching
- Shared memory architecture - system memory / frame buffer
- Integration of memory / graphics / video DMA controllers



The Role of Core Logic in Multimedia PCs

System --> Motherboard

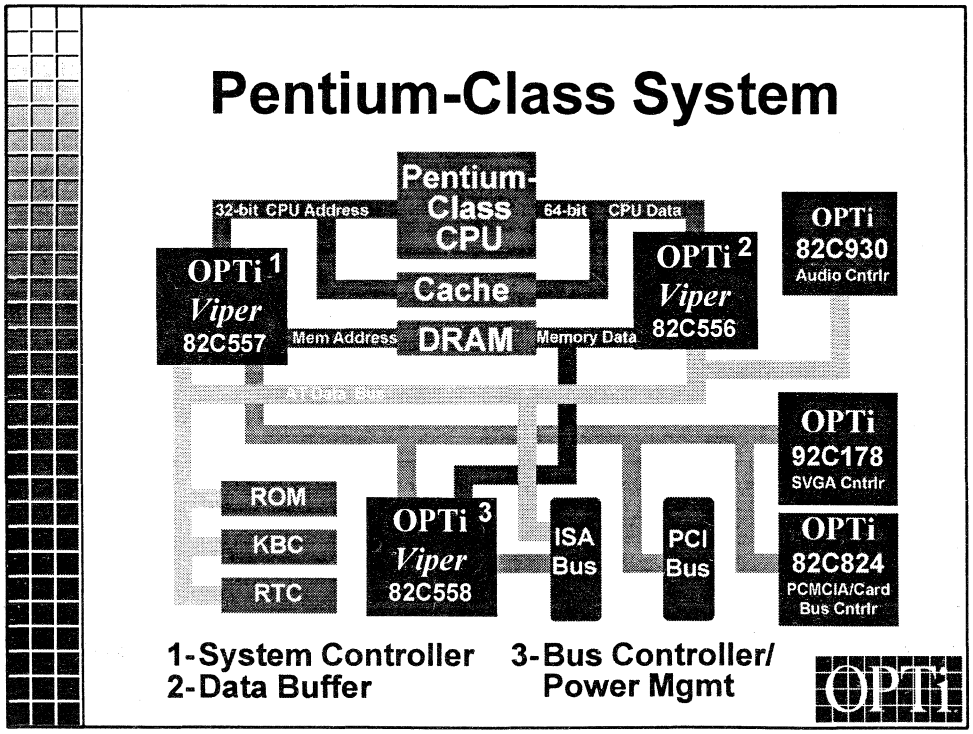
Multimedia Performance

Core Logic

Bus Standards

Integration

OPTi



Pentium-Class System

Pentium-Class CPU

32-bit CPU Address

64-bit CPU Data

Cache

DRAM

Mem Address

Memory Data

AT Data Bus

OPTi[1] *Viper* 82C557

OPTi[2] *Viper* 82C556

OPTi 82C930 Audio Cntrlr

OPTi 92C178 SVGA Cntrlr

OPTi 82C824 PCMCIA/Card Bus Cntrlr

ROM

KBC

RTC

OPTi[3] *Viper* 82C558

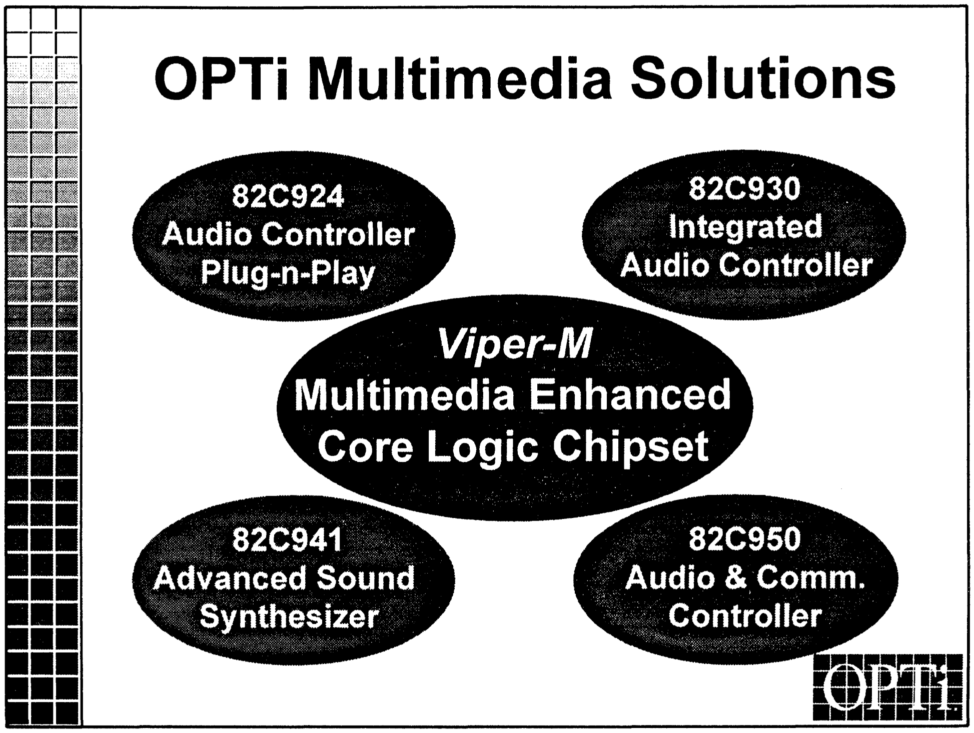
ISA Bus

PCI Bus

1-System Controller
2-Data Buffer

3-Bus Controller/
Power Mgmt

OPTi



OPTi Multimedia Solutions

82C924
Audio Controller
Plug-n-Play

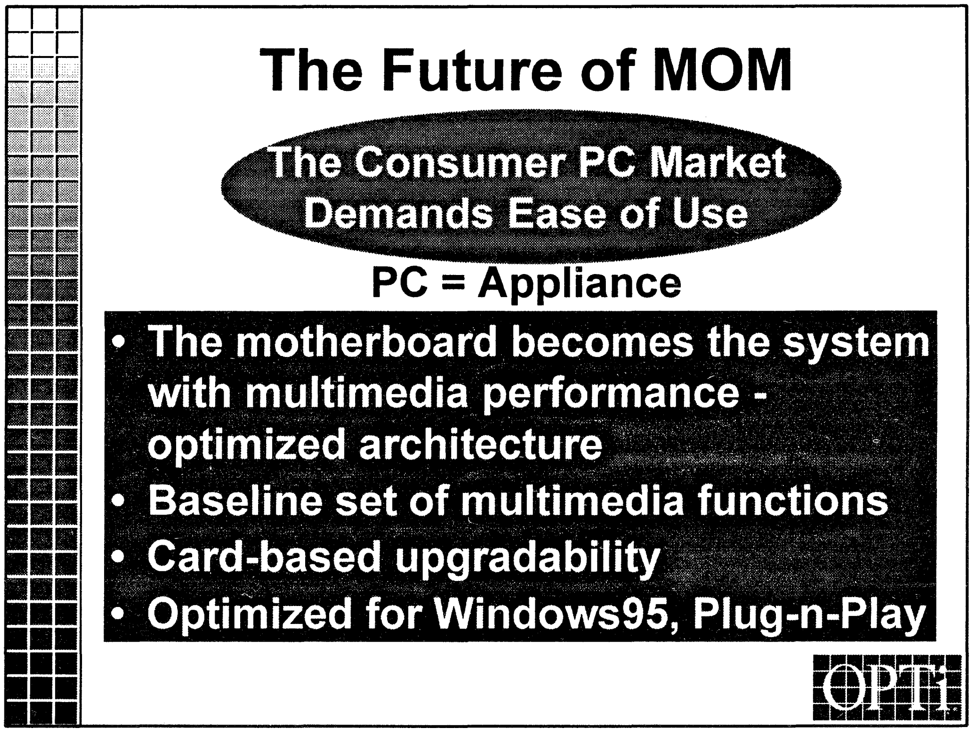
82C930
Integrated
Audio Controller

Viper-M
Multimedia Enhanced
Core Logic Chipset

82C941
Advanced Sound
Synthesizer

82C950
Audio & Comm.
Controller

OPTi



The Future of MOM

The Consumer PC Market
Demands Ease of Use

PC = Appliance

- The motherboard becomes the system with multimedia performance - optimized architecture
- Baseline set of multimedia functions
- Card-based upgradability
- Optimized for Windows95, Plug-n-Play

OPTi

Next Generation Chipset Performance

Dale B. Jorgensen
Applications Engineering Manager
PCI Components Division
Intel Corp.
1900 Prairie City Rd.
Folsom, CA 95630

The Intel Triton™ chip set represents a new generation in PCI chip set performance. For additional information about Intel's PCIsets, call a local sales office or the Intel Literature Center at 800-548-4725 (in the U.S. and Canada).

Implementing High-Speed LANs on PC Buses: Fast Ethernet Meets PCI

Dr. Charles R. Anderson
President
Cogent Data Technologies, Inc.
175 West St., P.O. Box 926
Friday Harbor, WA 98250
(360) 378-2929/2882 (fax)

The rapid evolution of local area network topologies poses an unprecedented challenge to traditional PC bus designs. Fast Ethernet, 100VG-AnyLAN, and ATM all push cable bandwidth to 100 Mbits/sec or more and result in sustained bus traffic of at least 10 MB/sec. With typical installations involving up to four simultaneous LAN channels in a single PC and with full duplex Fast Ethernet doubling these numbers, users are starting to need up to 80 MB/sec bus bandwidths just for their LAN peripherals.

Clearly traditional bus designs cannot begin to keep pace. With ISA bus speeds clocking a maximum of 4 MB/sec and even EISA realistically limited to 16 MB/sec, there just isn't enough bandwidth to handle new LAN topologies.

PCI offers the only realistic solution for high-speed network designs. With a theoretical bus bandwidth of 132 MB/sec, designers have a reasonable platform even for multiport Fast Ethernet designs. This presentation will cover some issues confronting designers of high speed LAN adapters for PCI bus systems and will analyze bus bandwidth utilization in traditional as well as PCI-based PCs. PCI to PCI bridge technology will be discussed as well, particularly in reference to some of the new multiport adapter designs becoming available in the market today.

PCI and the LAN

Tom Caldwell
Product Manager
Rockwell Network Systems
7402 Hollister Av.
Santa Barbara, CA 93117-2590
(805) 562-3164/968-6478 (fax)
tcaldwell@rns.com

This talk will cover how network adapters utilize the PCI Local Bus to address needs for a variety of applications, including graphics, multi-media, and client/server applications.

Host Interface Design for ATM LANs

Michael H. Benson
ASIC Design Engineer
Fore Systems, Inc.
174 Thorn Hill Road
Warrendale, PA 15086-7535
(412) 772-6600/6500 (fax)
mhb@fore.com

ATM (Asynchronous Transfer Mode) is the newest and most talked about data networking standard. It offers high throughput, scaleable bandwidth, quality of service guarantees, support for multiple data traffic types, transparent interface to the WAN, and well-defined open standards. Along with these new capabilities has come the challenges of designing system bus interface that will deliver the required performance. With ATM, the network interface can no longer be considered as a low-speed peripheral. Currently, the most common ATM physical interface to the desktop, SONET OC3c, requires up to 35 MB per second of system bus bandwidth. In the near future, SONET OC12 will require 140 MB per second. Low system latency and efficient interface to the host's main memory are crucial to reducing the cost of delivering many high-speed connections through a single ATM interface. This paper will briefly describe ATM, the challenges of ATM applications, and the requirements that ATM network adapters place on the system bus.

Performance Implications in Designing ATM Adapters for the PCI Bus

Gary Kidwell
Senior Hardware Engineer
Interphase Corporation
13800 Senlac
Dallas, TX 75234
(214) 919-9000/9200 (fax)

The emerging ATM standard requires much higher bandwidth than traditional networks such as Ethernet and Token-Ring. This talk will discuss the performance implications of the various choices in designing ATM adapters for the PCI bus.

PCI BUS ATM ADAPTER DESIGN

Subbu Ganesan and Jim Hora
ZeitNet, Inc.
5150 Great America Parkway
Santa Clara, CA 95054 USA

ABSTRACT

ATM is a cell based circuit switching technology that is rapidly becoming the network technology of choice. Standards have been ratified which ensure interoperability among vendors. In this paper, we will give a brief overview of ATM and its network capabilities. Then we will focus on the PCI Bus specification and analyze its capabilities and advantages for processing ATM. We will illustrate the system analysis necessary for an ATM adapter design using the PCI Bus and discuss the advantages of using the host memory as the ATM data memory. Lastly, we will discuss some of our experiences with PCI bus computer systems and its future potential for providing high performance ATM solutions.

1.0 INTRODUCTION

The acceptance of ATM in the marketplace has occurred rapidly because ATM has several advantages over other networking technologies.

ATM provides for constant bit rate services that require real time processing capabilities, including low latency, Quality of Service (QOS) parameters, and traffic management. These capabilities are possible because ATM is a fast cell switching technology based on a fixed length 53 byte cell. By switching short fixed length cells over a switched fabric, networks can provide real time processing capabilities. ATM allows variable bit rate (VBR) services to be combined with constant bit rate (CBR) services to provide efficient usage of the network.

The ATM has developed standards for multiple bit rates including standards with over 1 Gbps data rates. These multiple bit rate standards simplify LAN/WAN connectivity over an ATM fabric by providing scaleable connectivity to higher bit rates.

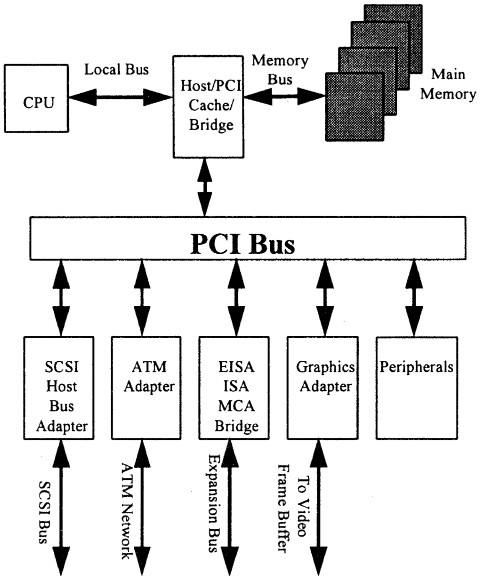
The ATM Forum has developed standards for signaling that allow end stations to setup a connection between them which defines the control parameters necessary for maintaining the performance required for real time processing. LAN Emulation, another standard developed by the ATM Forum, provides for compatibility with legacy LANs such as Ethernet and Token Ring.

ATM is a connection oriented technology that allows for Quality of Service (QOS) parameters such as bandwidth, latency, and priority to be specified at the setup of the connection. To handle the problem of congestion, the ATM Forum has implemented traffic management control standards to provide an even more robust solution.

This paper will focus on the PCI Bus architecture and how it will support a high performance ATM adapter design. In Section 2.0, key parameters of the PCI bus architecture that relate to an ATM adapter design will be analyzed. In Section 3.0, we will analyze an ATM adapter design and discuss the system level design decisions that are required during this process. We will show how the PCI bus architecture satisfies the ATM requirements of high bandwidth and low bus latency for real time processing as well as highlighting the PCI Bus Plug and Play capability. In section 4.0, we will discuss our experiences with the PCI Bus and highlight the future growth path available for the PCI Bus and ATM networking.

2.0 PCI BUS ARCHITECTURE

Let's look at a typical PCI bus computer system.



Typical PCI Bus System with ATM Adapter

The PCI Bus is a high performance bus that is well suited for transferring data between peripherals, adapters, and other bus backplanes. The types of adapters that are supported by the PCI bus control include adapters for external memory accessing (typically a SCSI adapter) and adapters for networking connectivity (including ATM). The PCI Bus supports expansion busses through the use of a bridge and other peripheral devices which include audio and video peripherals.

The PCI bus is usually connected to the host CPU and main memory through a bridge device that controls the data transfers between the CPU, Cache and Main Memory. This bridge also provides the major interface and controls the data transfer between main memory and all of the other devices on the PCI bus.

The PCI Bus architecture has addressed several of the shortcomings of other bus architectures. By addressing these issues, the PCI bus architectures enables high performance networking solutions. High performance ATM networking solutions require specific bus performance capabilities, such as bandwidth, latency, and burst data transfer as well as containing configuration capabilities to allow for Plug and Play.

The key features of PCI bus architecture that make it an attractive bus for an ATM adapter solution are:

Raw Bandwidth
Low Latency
Burst Capability
Processor Independence
Plug & Play
Growth

Let's analyze these key features of the PCI bus architecture. Then in Section 3, we will apply these key parameters in a system design analysis of an ATM adapter for the PCI bus.

2.1 Bus Bandwidth

Bus bandwidth is extremely important to not only networking performance but also system performance. The PCI Bus is capable of high performance data transfer through its high bus bandwidth capacity. To calculate the maximum PCI bus transfer rate, we multiply the bus clock rate by the number of bits on the bus and then divide by the number of clock cycles it takes for each data transfer (It is 1 for the PCI bus):

Clock Rate = 33 Mhz
Bus width = 4 bytes = 32 bits
Max. transfer rate = 133MBytes/s= 1.04Gb/s

From this data, we can see that the maximum transfer rate of over a gigabit per second is much greater than the transfer rate required for ATM. (The SONET 155 Mbps ATM rate requires 134 Mbps data rate. The data rate difference is due to the amount of SONET overhead data which is not sent over the PCI bus) However, just having the available bandwidth does not guarantee good ATM performance. This will be discussed in the ATM adapter system design in Section 3.

2.2 Variable Burst Size

A burst transfer is comprised of and address phase and one or more data phases. The burst size is the number of data phases multiplied by the number of bytes on the PCI bus (4 for 32 bit PCI bus) that occur between the bus master and the target. The PCI Bus specification allows for a variable burst size and does not restrict the burst size. The upper limit on the burst size is set depending on the system load, the latency timer, and what the target can tolerate. This means that in a lightly loaded system the burst size could be very large. In a system with several Bus Masters, the bus bandwidth is distributed among the Masters.

2.3 Bus Latency

Bus latency is the time between a master requesting access to the PCI bus and the completion of the first data phase. Therefore, bus latency is a very important requirement necessary to provide low cost/high performance ATM solutions.

There are several factors that contribute to bus latency. The overall bus latency is comprised of three parts: arbitration latency which is the time the Master waits after asserting REQ# until it receives GNT#, bus acquisition latency which is the amount of time the device waits for the bus to become free after GNT# has been asserted, and target latency which is the amount of time that the target takes to assert TRDY# for the first data transfer.

The PCI handshake is designed to be as a worst case scenario - 2 clock cycles. For time critical transfers, the PCI specification provides Latency Timers which all Masters have to implement. When a latency timer expires, the Master has to release the bus after the current data transfer, if the Grant is removed. This allows time critical data transfers to gain access to the bus in a timely manner.

2.4 Plug & Play

Plug and Play is a concept where any adapter or peripheral card can be plugged into the bus and it will automatically be configured and work (Play) in the system without any problems. The PCI Bus specifications provides for an automatic configuration of the hardware in the system, by allowing the operating systems to set the interrupt levels, base address, latency timers and obtain directly from the hardware any information required by the driver / OS. This eliminates conflicts between boards in the system as well as eliminating the need for jumper headers on a board that were common in the past and caused many problems during installation. The configuration registers allow the operating system to determine the type of cards in the system, gather information on the requirements of the hardware and its setup parameters, like the bus latency timers, inorder to optimize system performance. Below is the information contained in the PCI bus configuration header.

Device ID		Vendor ID		00
Status		Command		04
Class Code			Rev. ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Registers				10
Reserved				28
Reserved				2C
Expansion ROM Base Addr				30
Reserved				34
Reserved				38
Max_Lat	Minx_Gnt	Interrupt	Interrupt	3C

Configuration Space Header

2.5 Processor Independence

Processor independence is when the operation of the system is not dependent on the type of processor used. This allows all devices on the PCI bus to operate in systems that have different processors, as long as they provide a PCI interface Bus.

2.6 Growth

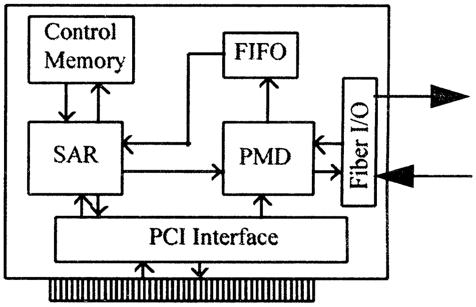
A high performance specification needs to address the needs of today as well as allowing for growth in the future. The PCI bus specification has been designed to grow with technology. This will be discussed in section 4.1.

3.0 A PCI BUS ADAPTER DESIGN

When designing an ATM adapter card for any bus specification, one needs to have their goals clearly established. As we set out to design an adapter card for the PCI bus, we had the following goals:

1. Be able to sustain full duplex line speed
2. Low cost (<$1000)
3. Operate in multiple software environments
4. Simple installation process
5 Operate on multiple PCI platforms

The PCI bus allowed us to meet these goals. To start the system design analysis process, lets take a look at an ATM system block diagram for the PCI bus.



PCI Bus ATM Adapter Block Diagram

The major functions that are needed for a PCI bus ATM adapter are the SAR[1], PMD[2], FIFO, Control Memory, PCI interface, and Fiber Transceiver. There is one functional block that is not included in this block diagram and that is data or packet memory. As we go through the system design for the adapter, it will become obvious that the PCI specification allows the adapter to transfer data directly to host memory with minimal buffering required on the adapter. This was necessary to achieve the cost goals for our adapter.

The following analysis identifies the requirements of an ATM adapter from an end station point of view.

[1] Segmentation and Reassembly

[2] Physical Media Interface Device

3.1 Bandwidth

ATM networks running on SONET OC-3 transfer data at 155 Mbits/sec which results in a bus bandwidth requirement of about 20MB/s (134Mb/s) on the receive side. To sustain line rate on the transmit side we require another 20MB/s[3].

Therefore, to sustain Line Rate for both Transmit and Receive, we need a raw bandwidth of 40 MB/s. As shown in Section 2.1, the PCI bus easily provides this raw bandwidth by design (>1 Gbps). The ATM adapter would only take up 30% of the raw bandwidth even during peak transfers.

3.2 Bus Latency

Predictable and low bus latency are very important for ATM adapters. If bus latency was not low and predictable, one would need to add data memory to the adapter or provide a large buffer. As mentioned in Section 1, applications running on ATM networks require guaranteed latency parameters. Isochronous (time sensitive) applications depend on predictable latency values.

In a system with PCI devices, the latency value is a configuration parameter that the adapter can request. The boot software determines the latency timer value based on the load in the system. For a given latency timer value, the maximum latency is fixed and predictable.

The PCI bus latency specification guidelines for PCI devices states that the typical latency is short (likely under 2usec and predictable. If for example the LT timer is set to 40, which is a typical value for the PCI bus, the maximum latency would be 1.6usec and the peak bandwidth would be 100MB/sec. This bandwidth is still above ATM bandwidth requirement.

The latency is more difficult to predict for existing PCI systems that have ISA or other expansion bus devices. This is because devices on the expansion bus do not comply with the latency requirements of the PCI. The PCI Specification suggests using 30usec as a worst case latency in such systems.

Using 30 microseconds as a worst case analysis of how long the ATM adapter would have to wait to receive control of the bus, we can calculate the amount of data that would need to be stored on the adapter. As we discussed in section 3.1, the bus bandwidth required for sustained line rate data transfer is about 20MB/s. Therefore:

FIFO Size = 30usec X 20 MB/s
= 600 Bytes

Therefore, even at line speed, only a 1K FIFO is needed for the worst case latency. For the typical case of 2 microsecond bus latency, a FIFO of less than 100 Bytes is necessary.

3.3 Burst

The SAR fetches one cell at a time from the host transmit packet memory and similarly assembles one cell at a time on the host receive packet memory. Being able to burst 48 bytes (The ATM cell size) across the PCI Bus helps in reducing the overhead and utilizing the bus more efficiently. The Host/PCI bridge chip sets limit the burst transfer size to 32 bytes during a read cycle and 16 bytes during a write cycle (this is discussed in section 4). Let's look at the bus cycle timing for a PCI read cycle for a 32 byte burst.

Action	Number of clocks
Bus Request to Grant	4
Address Cycle	1
Target Delay	10
Data Transfer (8 Words / 32 Bytes)	8
Turn-around & Idle	2
Total	25

PCI Burst Read Cycle

The PCI burst read cycle uses 25 clock cycles to transfer 32 bytes of data. This provides an effective bandwidth of 340 Mbps even with the bus latency added in. The bus cycle timing for a PCI burst write cycle is:

Action	Number of clocks
Bus Request to Grant	4
Address Cycle	1
Target Delay	2
Data Transfer (4 Words /16 Bytes)	4
Turn-around & Idle	2
Total	13

PCI Burst Write Cycle

[3] Note that ATM is "Full Duplex" unlike Ethernet

The PCI write cycle uses 13 clock cycles to transfer 32 bytes of data. This provides an effective bandwidth of 328 Mbps even with an allowance for bus latency

The PCI bus specification allows for bursting large data transfers or packets across the Bus, since the packet size is typically between 4KB to 10KB. Since the PCI bus is a low latency/high bandwidth I/O bus, it is obvious that the Master Latency Timer will expire and the device would have to give up the bus. If one device tries to burst large data packets across the bus, it would monopolize the bus. This would negate on of the benefits of the PCI bus and reduce the performance of other devices, like audio and video controllers on the bus. The latency timers allow several isochronous adapters/peripherals to be on the bus at the same time while not impacting the performance.

3.4 System Decisions

After analyzing the bandwidth, latency, and burst capabilities of the PCI bus, we can now go back and look at the goals for the design of the PCI bus ATM adapter and determine whether they were met. From the analysis in the burst section, we can see that both the PCI bus read cycle and write cycle support ATM line speeds which fulfill goal number 1 of full duplex sustained line rate.

The second goal was the one that required the most analysis. The analysis of the bus latency for the PCI bus shows that packet memory is not needed on board and that only a 1K FIFO is needed to buffer the data and still sustain ATM Line Rate. There is a large cost savings by using the host memory as the packet memory. The amount of packet memory needed for ATM is calculated in the table below. This assumes that each virtual channel (VC) requires 10K buffer space (unidirectional transfer only), which is the maximum IP over ATM packet size.

Number of VCs	Memory Required	Memory Cost
32	320KB	$30.00
64	640KB	$60.00
100	1MB	$100.00
1K	10MB	$1000.00

Packet Memory Cost

From this table, we can see that the amount of memory needed grows quite rapidly and is still considerable even for a low number of VCs. For full duplex operation, the amount of memory would need to double to support both transmit and receive. It is important to note that the use of packet memory on the adapter does not change the amount of host memory required. The same amount of host memory is determined by the size of the packet and the number of VCs and is independent of the amount of memory on the adapter. The PCI bus specification allows an ATM design to be done avoiding the excess cost of packet memory. By designing the adapter without packet memory on the adapter, we were able to meet the cost goal.

Goal number three was to operate in multiple software environments. This was accomplished during our testing with the PCI bus adapter. the PCI bus supports multiple operating systems including Windows NT, Novell Netware, Windows for Workgroups, NDIS, and more. We have been able to operate with these multiple operating systems which satisfies goal number three.

In Section 2.4, we discussed the configuration capabilities of the PCI bus specification. Using the configuration space header data, we have been able to simplify the installation process and meet goal number 4.

Goal number five, which was to operate on multiple platforms, was met and will be discussed in Section 4.

4.0 ZEITNET'S PCI EXPERIENCE

The PCI bus is becoming very popular among computer vendors which has resulted in a lot of computer manufacturers that are supporting the PCI bus specification. Inorder to do thorough testing of our PCI adapter design, we had to test our adapter in several variations and types of PCI computers. We tested our adapter on several different PCI machines including Compaq, Dell, DEC-PC, Gateway, Hewlett Packard, Micron, NEC PC and various clones. Each of these systems had other PCI devices on the system which allowed us to test a variety of typical systems. As one can imagine when testing multiple computer vendors, we found that not all PCI bus computers acted the same and discovered several interesting items during our testing of the ATM adapter. Some of these items satisfy the PCI specification, but may impact system throughput depending on planned use of the PCI specification. Here are some of our findings:

1) There are two manufacturers of Host/PCI bridges that supply most of the computer vendors. They both behaved similarly, but were not quite what we expected. Both Host/PCI Bridge chip sets limits the transfers to host memory to 32 bytes for a burst

read cycle and 16 bytes for a burst write cycle. This is due to the buffers provided in these host bridges for the PCI Bus. For ATM, the impact is minimal, but it does require the adapter to transfer a cell in two bus transfer (one transfer of 32 bytes and a second transfer of 16 bytes) to the host memory. We expect the new Host/PCI bridge chips to provide for larger transfers in the future.

2) Some of the BIOS software do not initialize the Interrupt Vector. This requires your software to update your interrupt vector. We expect most computer vendors to incorporate this in the future.

3) None of the existing systems that we tested with have implemented the latency timer. The latency timer allows for a more predictable bus latency and we believe that it will be implemented in the future on most systems which will improve the PCI performance.

4) The Host/PCI bridge contain extra control bits that have to be chosen carefully. Your software needs to ensure that these bits are chosen correctly and can not assume that the BIOS has done everything properly.

5) All systems do not enable parity. This requires you to process parity differently on some systems.

6) There are two mechanisms for identifying the devices in the system. Most systems use mechanism 1 to identify the hardware devices, but mechanism 2 seems to be coming more popular and will probably be used more in newer systems and in the future.

7) The target delay for the PCI burst read cycle was 10 clock cycles. We expect this to improve in the future and increase the performance of the PCI bus.

As you can see, you can not assume that all PCI bus computers will act the same. Your software needs to be versatile and monitor the system to ensure that all processing is done properly. Fortunately, there were no major problems and the PCI bus performed very close to expectations.

4.1 Future for PCI Bus and ATM

With any good technology or specification, future growth is an important consideration. The PCI bus specification is designed to grow with technology. There are several upgrades that are possible that will improve the ATM performance on the PCI Bus. These changes include;

1) The current PCI Bus is 32 bits wide. The PCI Bus allows for expanding the data bus to 64 bits. This change alone will double the maximum throughput.

2) The clock rate for the PCI specification is for 33 Mhz. The next generation of PCI Bus will allow for clock rates of 66 Mhz. This will also double the maximum transfer rates.

These two changes alone will provide for a maximum transfer rate of over 4Gbps. With the improved performance of the PCI Bus and the continued processing power growth, the possibilities for higher bandwidth [STS-12: 622Mb/s SONET] ATM networking on the PCI Bus are possible.

5. CONCLUSION

The PCI bus architecture is ideally suited for high performance networking applications. The key PCI bus features are:

Raw Bandwidth
Low Latency
Burst Capability
Processor Independence
Plug & Play
Growth

These key features allow users to create applications that utilize the power of ATM. But even more important is that the PCI bus allows one to design an adapter that is cost effective, which is essential to expanding high performance networking solutions for the end station.

It will now be up to the application vendors and users to create the next generation of products that can utilize the true power of ATM and the PCI Bus.

Subbu Ganesan: As a hardware architect for ZeitNet, he has worked on both the Sbus and PCI bus ATM adapter architectures. He has over ten years experience in high performance system design. His recent interests includes high speed bus interface design, ATM and other emerging LAN technologies.

PCI Latency and Bandwidth Issues for Network Adapters

Alan Deikman, alan@znyx.com
ZNYX Corporation, Fremont, California

Abstract

One of the basic applications of PCI is to provide an interface for LAN adapters that connect computer systems to networks. The PCI LAN adapter is quickly displacing the legacy bus adapters of the same type (i.e. PCI Ethernet instead of ISA Ethernet, etc.), because the higher transfer speed of the PCI bus provides more efficient data movements, reducing system overhead.

LAN adapters are characterized by having to interface a constant bit-rate data stream, such as a 10 megabit/second Ethernet or 16 megabit/second Token Ring, to a CPU/Memory system that is not always available at any given instant. This means that data being received has to be queued, or data transmitted may has to be pre-fetched. If the PCI bus latency is not adequately taken into consideration, data underrun and overrun conditions can occur.

PCI happens to be arriving into common use at a time when network technology is making a quantum increase in speed. A vast majority of networks installed before 1995 were installed with either at 16 or 10 megabits/second technology. For 1995 and beyond, there is a clear trend toward 100 megabit/second and faster LANs, with IEEE 802.3 fast ("Fast Ethernet"), IEEE 802.12 ("100BaseVG-AnyLAN"), FDDI and ATM. Since faster networks will cause more latency problems than slower networks, it is important for PCI system and peripheral designers to pay attention to these issues.

LAN Adapter Architecture

PCI provides two basic modes of data transfer: 1) target/slave mode, and 2) bus master. The method of choice used for LAN adapters is bus master, which allows higher speed burst transfers and, more importantly, the ability to transfer data without the involvement of the host CPU.

As a result, the typcial PCI LAN adapter has an architecture similar to that shown in Figure 1. For simplicity the components implementing target/slave mode registers for device configuration and control have been omitted.

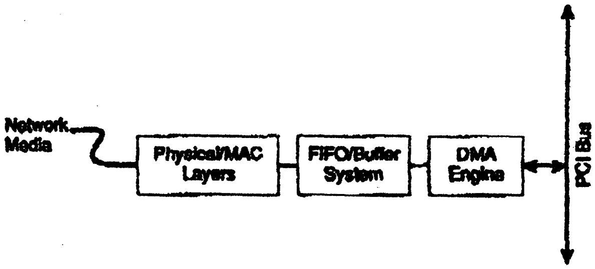


Figure 1 - Typical PCI LAN Adapter

Because a PCI based system is expected to have a bandwidth that is high relative to that of the Network Media, (even for high speed networks) most PCI LAN adapters do not implement independent, randomly accessed buffer memory. This saves on the overall board cost.

The FIFO/Buffer system provides an adaptation between the lower speed, constant bit-rate data flow of the Network Media to the higher speed, variable bit-rate data flow of the PCI bus.

During a transmit operation, the FIFO must be filled with data from the PCI bus faster than the Physical/MAC Layer device consumes it. If it does not, a FIFO underrun occurs and the transmit operation must be aborted.

During a receive operation, the FIFO must be emptied by the PCI bus faster than the Physical/MAC Layer fills it. If this does not happen, a FIFO overrun occurs and the incoming data packet is corrupted.

The problem will be compounded if the LAN adapter supports full duplex operation, providing a separate transmit FIFO and a receive FIFO. Underruns and overruns will occur under one of two conditions:

1) The bandwidth of the Network Media is larger than the bandwidth on the PCI bus that is available to the LAN adapter.
2) The time that it takes for the LAN Adapter to acquire and transfer data on the PCI bus. This is called the PCI Latency.

The larger the FIFO, the less likely an underrun or overrun event will occur. It is tempting to simply design a device with the FIFO as large or larger as the maximum size packet and this has been done with at least one design, and thereby guaranteeing that any PCI latency condition can be accommodated. However, the trade-off is cost; a 1514 byte FIFO for a full length Ethernet data packet can consume 80,000 or more gates on a standard-cell ASIC. In addition, the LAN board design becomes a store-and-forward design, which imposes a network transmit/receive latency equal to the size of the data packet. Competing designs that do not impose this network latency will benchmark higher.

One of the principal design questions of the LAN adapter designer, then, is how large does the FIFO need to be? The optimal choice will be the smallest size that will never allow an underrun or overrun to occur in the worst case latency scenario for the supported systems.

PCI Bandwidth

Before looking at latency, a quick overview of PCI bandwidth is helpful. PCI is widely advertised as being capable of "132 megabytes per second," which has led to the general assumption that this is the bandwidth of PCI. The figure is derived from a 33MHz bus clock (30 nanoseconds), with 32 bits of data being transfered each cycle.

The PCI Specification allows for a bus clock as slow as 25mhz, which will result in a lower "rating" of 100 megabytes per second.

In practice, the actual amount of data that can be moved over PCI is somewhat smaller than this theoretical maximum, since a certain amount of overhead is inherent in the design. The sources of this overhead are:

- Bus arbitration overhead - cycles are lost when the PCI bus arbiter switches from master to master.
- PCI burst length - data transfers are broken up into bursts, each with an address cycle. The shorter the burst, the more address cycles will consume the bus.
- Bus turnaround - each PCI read burst requires a turn-around cycle to allow the bus to settle between alternate drivers.
- Wait states imposed by the PCI target device(s).
- Wait states imposed by the PCI master device(s).

Figure 2 illustrates how PCI Burst Length can affect PCI bandwidth. For simplicity, the figure disregards the possibilities of wait states and bus arbitration.

Assuming 28 bytes are to be transferred in 7 data cycles, the PCI bus will actually take 19 clocks to complete the transfer in a bus-master read operation. With a 33MHz clock, this is at a rate of 35 megabytes/second, or only 26% of the nominal 132 megabytes/second theoretically possible! (Still faster than EISA, but considerably less than what the customer probably expected.)

Increasing the PCI burst length to 8 greatly improves matters. Here the efficiency of the data transfers climbs to 71% for reads of 28 bytes and 87% for writes of 28 bytes. Longer transfers and larger PCI burst lengths will gradually increase the efficiency of the bus.

Clearly, the silicon designer of the LAN adapter should make sure to maximize these parameters of the design, since the remainder of the bandwidth has to be shared between all the bus-masters contending for the use of the bus.

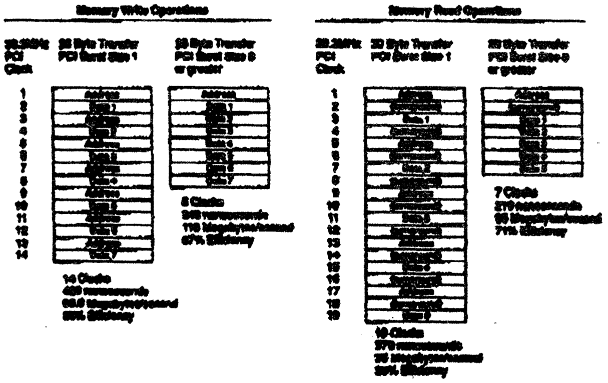


Figure 2 - Effect of PCI Burst Length on Bandwidth

It is important to note that if a bandwidth problem exists in a system, no amount of FIFO space in the LAN adapter can prevent overrun conditions. (Underrun can be eliminated with a store-and-forward FIFO.) Although problems can be postponed on some scale, it only requires a data burst long enough before a packet is lost.

PCI Latency

The design of PCI allows the configuration of systems with predictable bus access latency. This is done by limiting the amount of time that each bus master device can own the bus, and then granting access to all requesting bus masters on a "fair" basis.[1]

The principal mechanisim for making sure that each PCI device receives a portion of the bus is the master latency counter.

Maximum Arbitration Latency in Clocks							
LAT Counter	Number of Bus-Masters Ahead in Arbitration 0	1	2	3	4	5	6
0	2	2	2	2	2	2	2
2	2	4	6	8	10	12	14
4	2	6	10	14	18	22	26
8	2	10	18	26	34	42	50
16	2	18	34	50	66	82	98
32	2	34	66	98	130	162	194
64	2	66	130	194	258	322	386

Table 1

PCI Latency is broken down into three components, as illustrated in Figure 3.

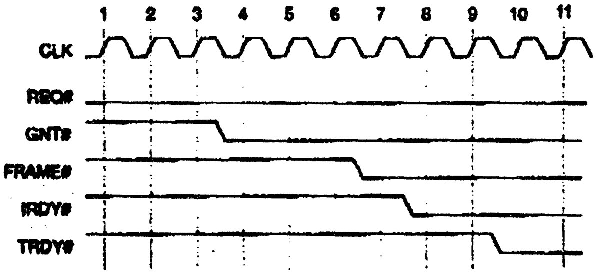


Figure 3 - Latency Components

Problems Observed in Real-Life

The PCI Specification 2.1, states "Inclusion of a standard expansion bus (ISA, EISA, or MC) makes latency prediction more difficult when a PCI agent is accessing an agent on the expansion bus."[2] This turns out to be an accurate prediction. Latency problems can occur for two reasons: 1) from the design of the legacy bus bridge, and 2) the latency of the legacy bus device itself.

NetWare and ISA video

One latency problem scenario has been observed using a 10Mbit Ethernet Adapter with a 256 byte transmit FIFO and a 256 byte receive FIFO. This device is able to tolerate a PCI latency of around 200 µsec, which is many times larger than that suggested by the PCI Specification.[3] The problem system was based on Novell NetWare, a standard 486/586 class PCI mother-board, the PCI Ethernet adapter, and an ISA video adapter.

The PCI Ethernet adapter is a bus-master device, which stores and fetches LAN data into and out of main memory according to instructions given it by the adapter's driver. The software driver is able to detect FIFO underrun or overrun incidents.

The NetWare monitor screen, when active, updates its display once each second. On a slow ISA adapter, this causes larger than normal latencies on the PCI bus each second which, if data is being transferred by the LAN adapter at that time, result in an underrun or overrun being reported by the driver. When the monitor screen is deactivated, no underruns or overruns appear under the same network load. The conclusion is that the wait states imposed on the PCI bus by the expansion bus bridge (on behalf of the video card) results in latencies longer than 200µsec.

Although this situation does not threaten the security of the user's data or, in most cases, noticably affect performance, most system purchasers find this condition unacceptable. The cure most often seen is to use a PCI video card (although high speed graphics is not normally associated with server applications).

Memory Subsystems

In many systems, the speed of the PCI bus is not the system bottleneck. Instead, the limitation is the ability of the host DRAM subsystem to source/sink data. A simple memory subsystem, composed of 36-bit SIMM, is often limited to a total capability of 40 megabytes/second, compared to the 100 megabyte and above data that can appear from the PCI bus.

As a result, a four-channel Ethernet Adapter, configured as four NICs beyond a PCI-PCI bridge, can overrun some systems.in the configuration shown in Figure 4.

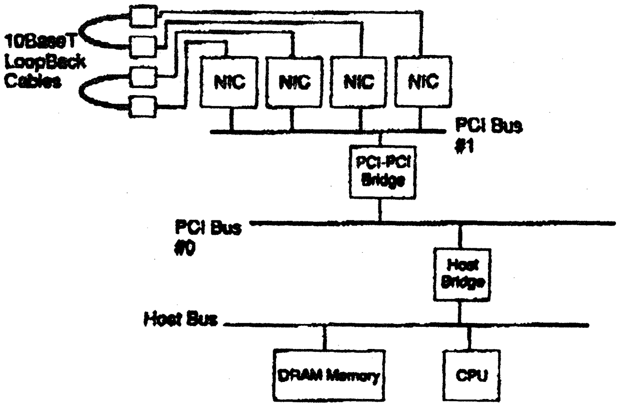


Figure 4 - Four Port Test Setup

In this test setup, each NIC is programmed to continuously transmit data packets stored in the DRAM memory. Each port transmits at 10Mbps. Since each NIC is also looped back to another NIC, each port is also continuously receiving at 10Mbps. So the overall payload data rate can reach a maximum 2.5 Megabytes/second per NIC, or 10 Megabytes/second for the total test setup.

Some mother-boards have been observed that are not able to handle this data load. Although these mother-boards tend to be the lowest-end product, it is an important demonstration that shows that PCI bandwidth does not necessary indicate data processing capability.

References

[1] Page 45, PCI Local Bus Specification, Review Draft Revision 2.1, October 21, 1994

[2] Page 52, PCI Local Bus Specification, Review Draft Revision 2.1, October 21, 1994

[3] Page 45, PCI Local Bus Specification, Revision 2.0

Technical Issues in MPEG/Graphics Integration

Larry Chisvin
Markcting Manager

Multimedia Products Unit
Western Digital Corporation
800 E. Middlefield Road
Mountain Vicw, CA 94043

chisvin_L@a1.wdc.com
phone:415-335-2614
fax:415-335-2515

Abstract

This talk will focus on the technical trade-offs involved whcn integrating MPEG-1 playback capability with a graphics controller. Various methods of connecting the video stream to the graphics function are reviewed, with an analysis of the advantages and disadvantages of each.

Mixed Media Streams over the PCI Bus

Brian Gibson
S3, Incorporated

Tomorrow's systems, especially home systems, will require an upgrade path that allows the user to add various multimedia functions. This must be low cost but still provide usable quality video and audio. The upgrade path could be via some type of standard connector or bus. The port/bus would carry various data types such as live audio/video (e.g. from a camera, VCR, laserdisk) and compressed audio/video (e.g. Indeo, Cinepak or MPEG files from a CD ROM). Several "standards" have been offered to provide this capability including the VESA Advanced Feature Connector (VAFC) and the Media Channel. While these two have their merits, neither has been very successful. The PCI local bus on the other hand is an effective upgrade path for multimedia data streams. Not only does it have sufficient bandwidth to handle mainstream multimedia applications, it is present in an ever growing number of systems. Therefore, at no additional cost, a user who purchases/owns a PCI based system has a built in expansion path. Following are some issues to consider when deciding on a multimedia platform.

- **Bandwidth requirements/issues for multimedia data streams**
  - Source image size and associated bandwidth requirements
  - Destination image size and associated bandwidth requirements
  - Multiple images (e.g. video conferencing)
  - Live video
  - MPEG/Indeo/Cinepak playback
  - Audio streams

- **Latency effects**
  - FIFO size
  - Arbitration priority schemes
  - Efficient use of bus

- **Number of upgrade slots**

- **Multiple functions on PCI bus**
  - PCI to PCI bridge (secondary PCI bus)
  - Busless interface to PCI device

- **Interrupt issues**
  - Allocation by system
  - Proper sampling of interrupt line
  - Interrupt routing/sharing

- **PCI vs VMC/VAFC**

Components and Adapters are available today that take advantage of the PCI bus for multimedia applications. There are adapters that bring live or pre-compressed video over the PCI bus to be displayed by a graphics subsystem. More and more, the PCI bus will be utilized for multimedia. Those who are designing PCI components and systems need to help ensure that this platform can keep up with the fast growing multimedia market.

Implementing High Performance3D Graphics on the PCI Local Bus

Neil Trevett
3Dlabs, Inc.
2010 N. First St., Suite 403
San Jose, CA 95131
408-436-3456

3D*labs*

Implementing High Performance 3D Graphics on the PCI Local Bus

Neil Trevett
VP Marketing

Topics Covered

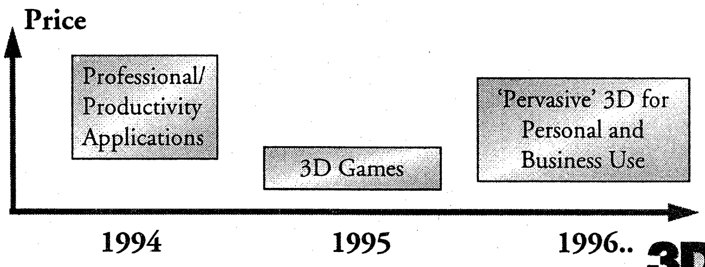
- Key technical issues for hardware designers and driver writers implementing accelerated 3D Graphics for Windows
  - Market requirements for 3D Graphics under Windows
  - The 3D Graphics pipeline, geometry and rendering, and the need for acceleration
  - Software architecture for acceleration
  - Hardware architecture for acceleration
  - Future directions for 3D under Windows

Background to 3D*labs*

- 3Dlabs designs and sells chips and technology for 3D graphics
- 3D is our core competency
  - 10 years experience of hardware and software design for 3D graphics
  - Hundreds of man years R&D Investment in 3D
  - A large patent estate in 3D graphics
  - A proven silicon design capability
- GLINT - our sixth generation 3D product
- Over 40 design-wins for GLINT
  - Mostly PCI boards for PCs

Opportunities for 3D Graphics

- Two separate market opportunities for 3D
  - Productivity and Games
- Early market driven by specialized Applications/Titles
- Eventually 3D will be Pervasive - 1997?
  - No longer driven by specific applications
  - 3D used in everyday applications and the user interface itself


Price
Professional/ Productivity Applications
3D Games
'Pervasive' 3D for Personal and Business Use
1994
1995
1996..

High Performance 3D Graphics on the PC is here

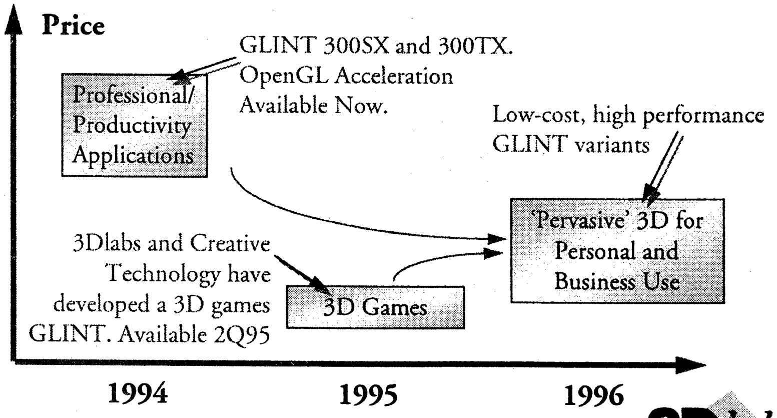
- All the components are now in place for high performance 3D on the PC
  - Pentium, MIPS and PowerPC class processors
  - PCI local bus
  - 32 bit operating systems - Windows 95 and Windows NT
  - Adoption of OpenGL by Microsoft
  - Fast 3D silicon, for example GLINT from 3Dlabs
- Application developers WANT to port to the PC platform
- Accelerated PCs are outperforming workstations running 3D intensive OpenGL applications TODAY

3D for Productivity and Games Have Very Different Needs

Professional Needs	Games Needs
High Quality	Low cost
Workstation performance	Interactivity
OpenGL	Games APIs
24 bit Z and color	16 bit Z and color
1280x1024 Minimum Resolution	640x480 Minimum Resolution
Windows, overlays, alpha buffer	Full screen, video, sprites
Competition: desktop Silicon Graphics	Competition: Sony PlayStation

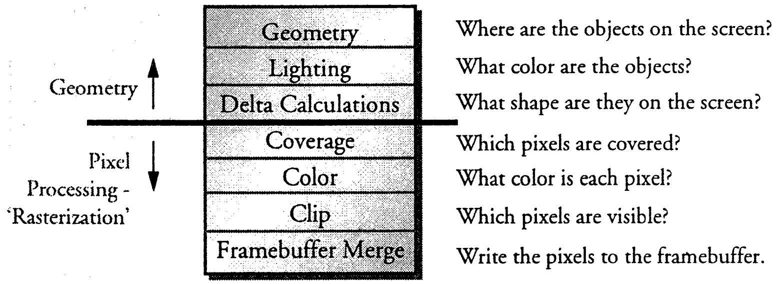
3Dlabs are Active in Both High-end and Games Markets

- Rest of talk concentrates on Professional / OpenGL market


Price
Professional/ Productivity Applications
GLINT 300SX and 300TX. OpenGL Acceleration Available Now.
Low-cost, high performance GLINT variants
'Pervasive' 3D for Personal and Business Use
3Dlabs and Creative Technology have developed a 3D games GLINT. Available 2Q95
3D Games
1994
1995
1996

The 3D Graphics Pipeline

- All 3D APIs have a similar architecture


Geometry
Pixel Processing - 'Rasterization'
Geometry
Lighting
Delta Calculations
Coverage
Color
Clip
Framebuffer Merge
Where are the objects on the screen?
What color are the objects?
What shape are they on the screen?
Which pixels are covered?
What color is each pixel?
Which pixels are visible?
Write the pixels to the framebuffer.

The Need for Acceleration

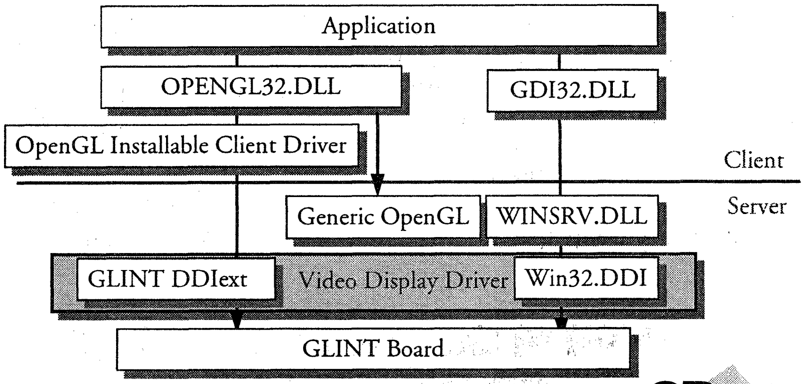
- Geometry
  - Functionality varies for each API
  - Floating point intensive
  - Large, complex code
  - Runs well on fast, standard processors
- Rasterization
  - Functionality common between APIs
  - Small, multi-field integer operations, bit shifting
  - Well suited to hardware implementation
  - Dedicated hardware provides x15 polygons/$ advantage over standard processors
- Use hardware for accelerating pixels not geometry

Overview of OpenGL

- Originally developed by Silicon Graphics
- Integrated by Microsoft into Windows NT and Windows 95
- The de facto standard for workstation class 3D applications
- Compute intensive geometry pipeline
- Extensive pixel processing functionality
- Strict conformance test - including consistency between rendering modes

Windows OpenGL Architecture

- By dynamically installing a .DLL, standard OpenGL applications can seamlessly use a 3D accelerator



Application

OPENGL32.DLL

GDI32.DLL

OpenGL Installable Client Driver

Client

Server

Generic OpenGL

WINSRV.DLL

GLINT DDIext

Video Display Driver

Win32.DDI

GLINT Board

Rasterization Performance

- 3Dlabs ships a fully optimized OpenGL .DLL for GLINT
  - Direct implementation from API to Silicon
- Advantages over using an intermediate Driver Layer
  - Higher performance - avoids an extra processing layer
  - Allows Optimizations, eg texture caching, command buffering
  - Allows Enhanced functionality, eg Multi-window double buffering
- A GLINT board typically provides 5-10x OpenGL speed increase on a P90 machine

3D-DDI - A 3D Driver Interface

- API and Chip vendors may decide to use 3D-DDI
  - A common interface layer between 3D APIs and chips
- A useful safety net for widespread availability of OpenGL and other APIs across hardware platforms
  - Precludes performance and functionality enhancements
- 3D-DDI is NOT an API for use by application developers!
- 3D-DDI will be available for GLINT
- GLINT currently support nine 3D APIs
  - All are implemented directly for maximum performance

Geometry Performance

- A 90MHz Pentium processes geometry for 100K visible OpenGL polygons/sec
  - GLINT can rasterize 300K polygons/sec
- The bottleneck is mainly floating point to integer conversions and function calls
- Faster processors help - MIPS and PowerPC are twice as fast, P6 is coming
- 3Dlabs Software Optimizations: Multi-threading for MP machines, Vertex Array Extension, Display List Caching
- Some GLINT boards will have hardware geometry acceleration - DSPs or PowerPCs
  - Need access to OpenGL source for a direct .DLL implementation

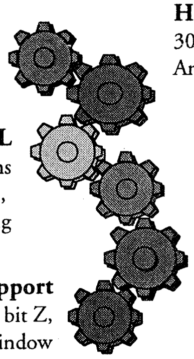
PCI Local Bus Design for 3D Performance

- The PCI interface can have a critical impact on graphics performance
- The use of DMA can double graphics performance whilst freeing the CPU for application processing
- GLINT provides comprehensive PCI DMA support
  - On-chip master capability and DMA controller
  - On-chip FIFO, with multiple address mappings to allow linear burst transfers into the FIFO
  - Command formats allow DMA of command and parameter stream

Specification for an Effective OpenGL Accelerator Chip

- Performance equal to mid/high-end desktop workstations
  - Professional users cannot afford to work any slower than on their workstations
- Complete OpenGL Rendering functionality in hardware
  - Applications use all the OpenGL rendering modes
  - Need hardware performance regardless of mode
  - Avoids software emulation of missing functionality
  - Any emulation must mimic hardware implementation
- Support for Workstation-class memory configurations
  - 32 bit color, 32 bit Z, overlays, double buffering
- PCI Master and DMA Capability

The GLINT Chip



Single Chip
Accelerates 3D and GUI

High Graphics Performance
300K Shaded, Z Buffered, Anti-aliased, Transparent Polygons/sec

100% OpenGL
All 3D Pixel Operations Accelerated in Silicon, including texture mapping

High Performance PCI Interface
Rev 2.0 compliant PCI Master, On-chip DMA

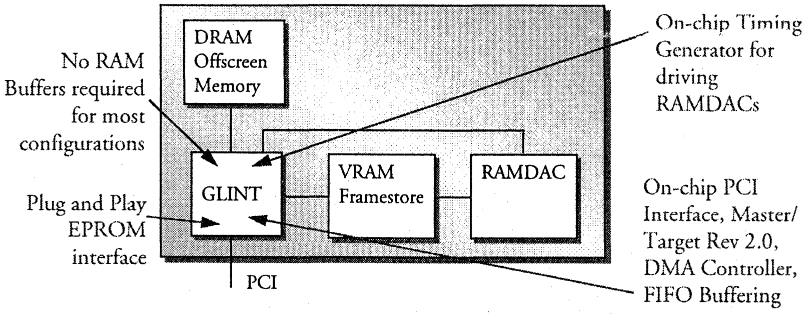
Flexible Memory Architcture
Upto 32 MBytes framebuffer and 48MBytes DRAM localbuffer

Deep Buffer Support
32 bit color, 32 bit Z, controllable per window

Workstation Class OpenGL Acceleration

3Dlabs

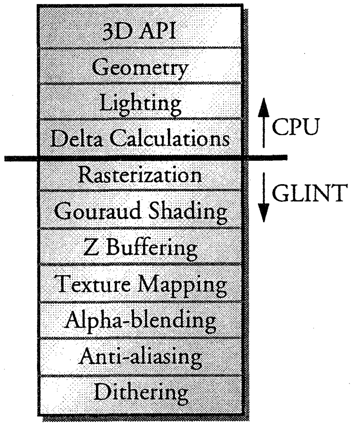
An Example 3D Accelerator Board Design Using GLINT



- Very low component count and board cost
- Direct connect to PCI, memory and RAMDAC
- PCI interface is full Rev 2.0 compliant

3Dlabs

GLINT Functionality - 100% OpenGL

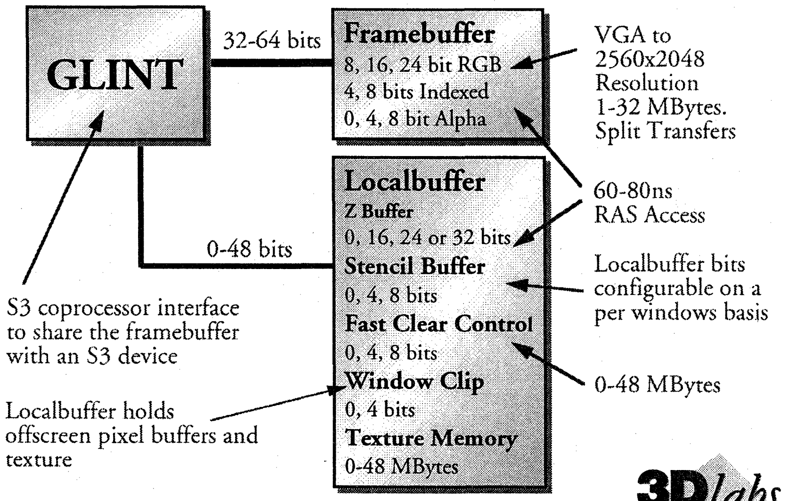


- Primitives
  Points, lines, triangles, rectangles, bitmaps
- Texture
  All OpenGL modes in silicon
  Including tri-linear mip-maps
  with per pixel perspective correction
- Anti-aliasing
  True 4x4 or 8x8 subpixel sampling
  Works for vectors *and* polygons
- Also
  Fogging, Alpha tests
  Windows and Scissor Clipping
  Stencilling and Stippling
  Depth cueing, Logic Operations

OpenGL is a Rasterization Superset of all other APIs - in turn GLINT can support ALL 3D APIs

3Dlabs

Memory Configuration Flexibility



3Dlabs

3Dlabs / S3 Alliance

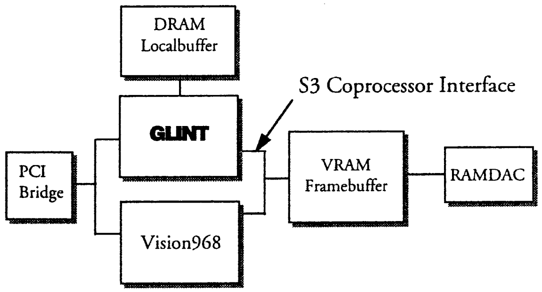
- S3 chips and GLINT can be used on the same board
- Maintains S3-based 2D driver and design investment
  - Seamless upgrade from 2D to 3D
  - All existing 2D drivers run without modification
- Scalable 3D solution
  - Low cost 3D designs using Trio64 + GLINT
  - High functionality using Vision64 + GLINT
- Video and 3D on one board
  - Vision968 has video support

3D*labs*

© Copyright 3Dlabs 1995 - PCI'95 Page 21
GLINT is a Registered Trademark and 3Dlabs is a Trademark of 3Dlabs Inc. Ltd.

GLINT and S3's Vision 968

- Shared VRAM Framebuffer
  - GLINT private localbuffer
- Vision968 performs all windowing and video functions
- GLINT accelerates all 3D within the window boundaries



3D*labs*

© Copyright 3Dlabs 1995 - PCI'95 Page 22
GLINT is a Registered Trademark and 3Dlabs is a Trademark of 3Dlabs Inc. Ltd.

Work in Progress for 3D under Windows

- Some workstation capabilities still lacking under Windows
  - Essential for many workstation applications
- Both GDI and OpenGL need support for:
  - Overlays
  - Double buffering in a window
  - Stereo
- 3Dlabs are working with Microsoft to define OS extensions

3D*labs*

© Copyright 3Dlabs 1995 - PCI'95 Page 23
GLINT is a Registered Trademark and 3Dlabs is a Trademark of 3Dlabs Inc. Ltd.

For More Information on GLINT

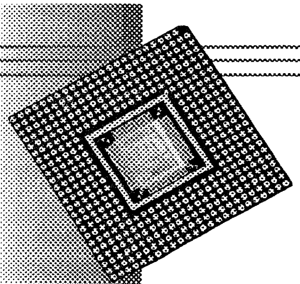
- Neil Trevett
- 3Dlabs Inc. is based in San Jose
- (408) 436 3456
- neil.trevett@3dlabs.com
-
- See us on the floor!

3D*labs*

© Copyright 3Dlabs 1995 - PCI'95 Page 24
GLINT is a Registered Trademark and 3Dlabs is a Trademark of 3Dlabs Inc. Ltd.

DECchip 21130 Integrated PCI Graphics and Video Accelerator

Frank T. Schapfel
Digital Semiconductor, DEC
77 Reed Rd.
Hudson, MA 01749
508-568-4122



DECchip 21130 Integrated PCI Graphics and Video Accelerator

Optimized solution for quality business video

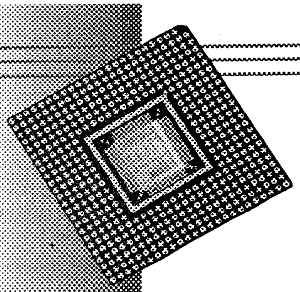
Frank T. Schapfel, Product Line Manager
Graphics and Multimedia
Digital Semiconductor

21130 Announcement

- High performance controller for business graphics and video applications
- First in a family of integrated PCI-bus display controllers
- Availability
  - Samples and Evaluation boards April '95
  - Production June '95
- Price
  - $34.10 @ 5Ku

Business Video Requirements

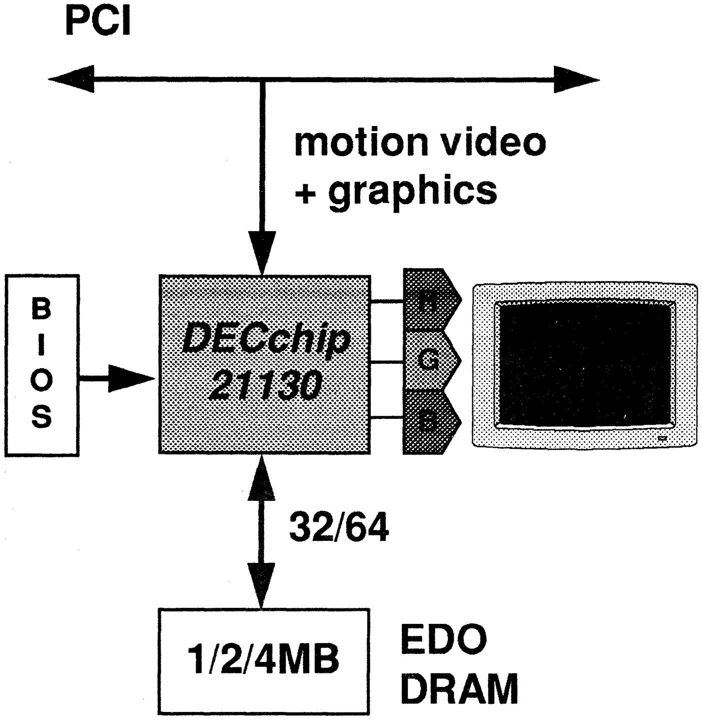
- Convergence of graphics and video on the desktop
  - Business Video
    - Multimedia presentations
    - Multimedia Authoring
    - Desktop Teleconferencing
    - Training and Remote Learning
  - Proliferation of CD ROM titles
    - Education and Entertainment
- Video performance dictated by end user expectation
  - Comparison to TV quality
- Jerky, blocky, postage-stamp video does not meet user expectations



What the 21130 delivers ...

- Outstanding graphics and video performance
- Optimized performance of SW video codecs
  - PCI Bus mastering
- AccuVideo^{TM}
  - Digital's Patent-pending imaging technology
  - Real time video at 30 frames per second
  - High resolution video
  - Multiple video window support
  - Simultaneous color quality for video and graphics

21130: Performance Graphics *AND* Motion Video



PCI

motion video
+ graphics

BIOS

DECchip 21130

R G B

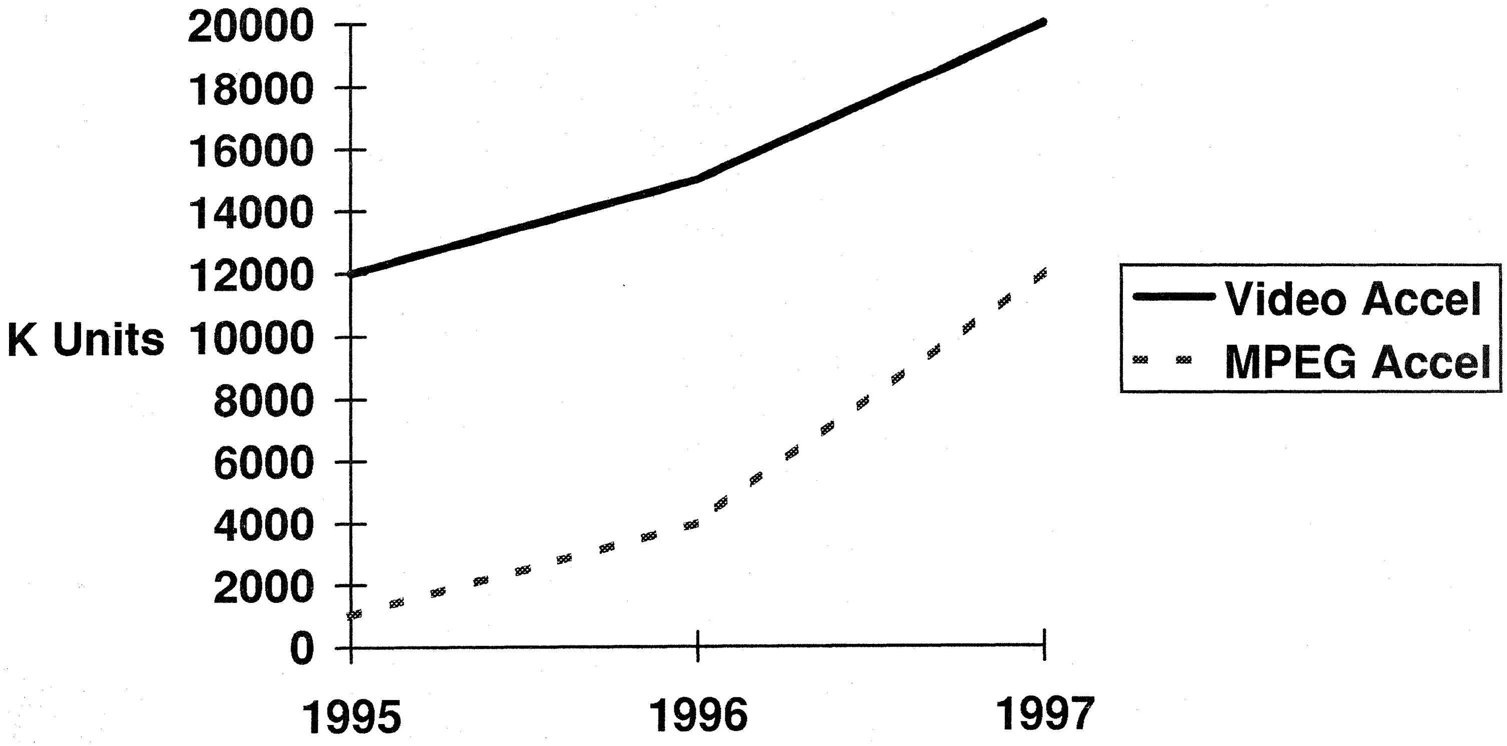
32/64

1/2/4MB

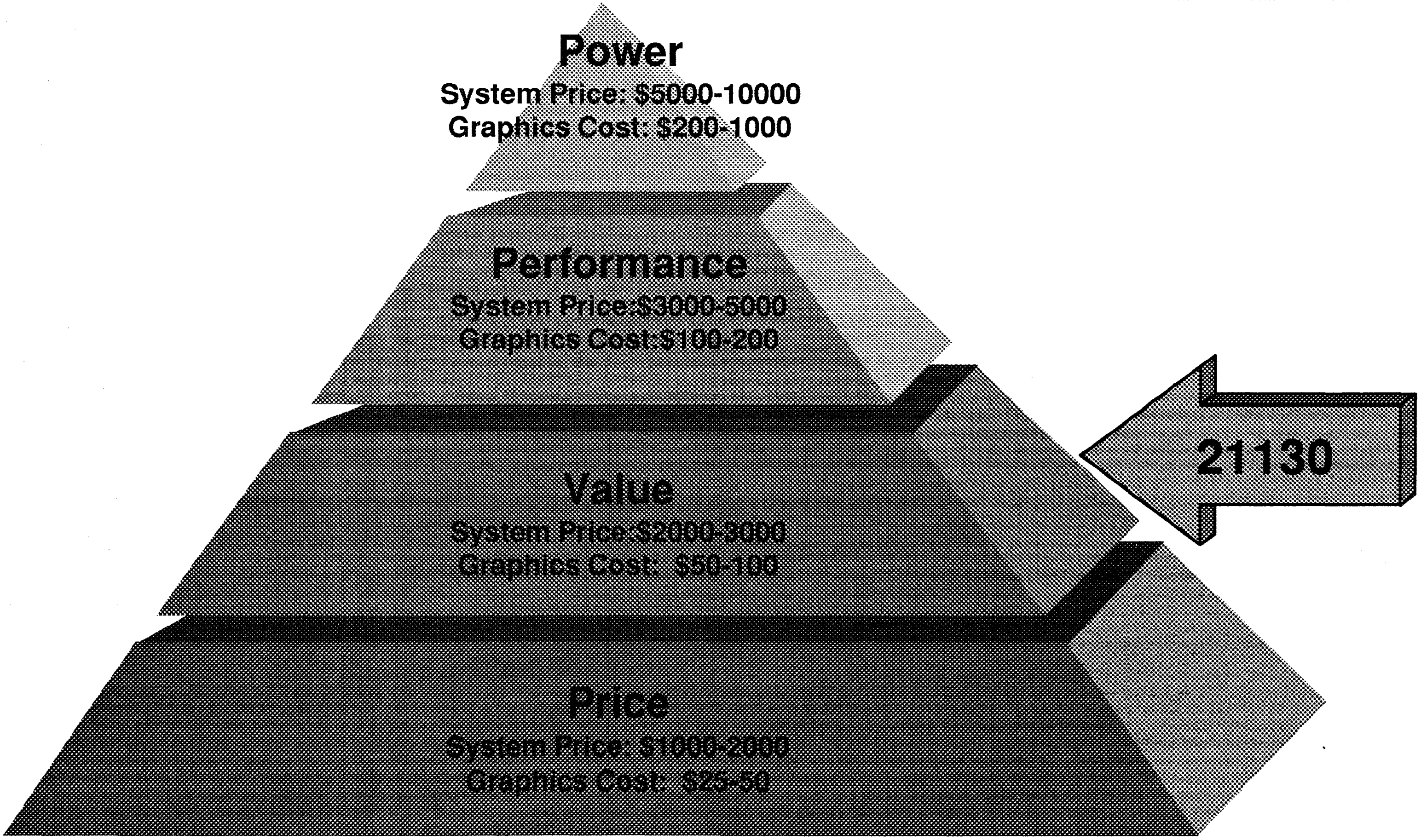
EDO DRAM

- Windows PCI Desktop
- Target applications
  - Multimedia presentations
  - Motion video playback
    - Real-time Indeo, Cinepak
    - Cost-effective MPEG-1
    - Games, "edutainment"
  - Windows acceleration
  - Upgrade to...
    - Video teleconferencing
    - Multimedia authoring
- Mainstream Pricing
  - ~$65 BOM cost for 1MB
  - ~$95 BOM cost for 2MB

PCI Video Acceleration Market Size



21130 Market Positioning

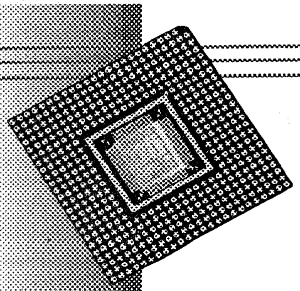


Power
System Price: $5000-10000
Graphics Cost: $200-1000

Performance
System Price:$3000-5000
Graphics Cost:$100-200

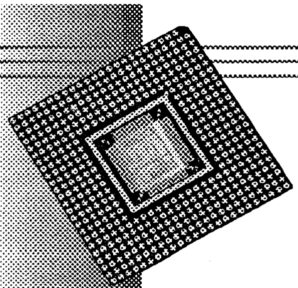
Value
System Price:$2000-3000
Graphics Cost: $50-100

Price
System Price: $1000-2000
Graphics Cost: $25-50



21130 Graphics Features

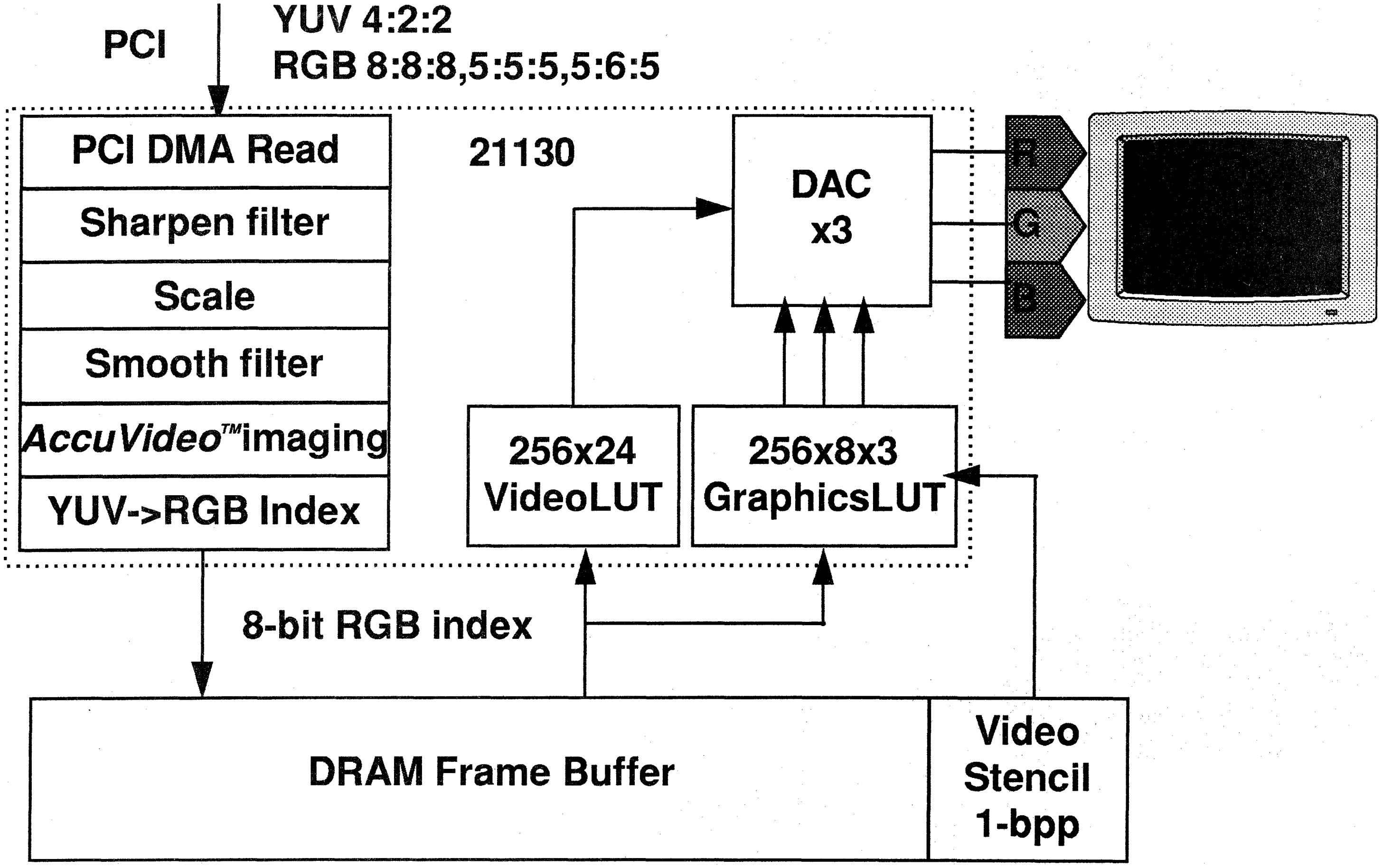
- 64-bit 2D Windows acceleration
  - BiTBLTs, line drawing
  - Full color solid and patterned fills
  - Color expansion for text and monochrome brush fills
  - 35M Winmarks 4.0 (Pentium-90 1024x768x256)
- Integrated VGA controller
- Integrated 135MHz RAMDAC
  - 3x256 color LUTs
- Integrated Phase-Locked Loop clock generator
- 32-bit PCI bus interface
  - Bus Master DMA read for fast BLTs to screen
- 32-/64-bit EDO DRAM frame buffer interface



The *21130* Difference
Best-in-class Video Acceleration

- *AccuVideo*™ imaging technology
  - 64K color quality
  - Unique imaging filters eliminate scaling artifacts
  - High resolution 1280x1024 in 2MB
  - Image scaling to arbitrary sizes
- Unlimited video windows
- Windows™ DCI compatibility
  - Monochrome or 8-bpp graphics overlay support on video
- YUV to RGB color conversion
  - No color palette conflict between graphics and video

21130 *AccuVideo*™: A Closer Look



PCI

YUV 4:2:2
RGB 8:8:8,5:5:5,5:6:5

PCI DMA Read

Sharpen filter

Scale

Smooth filter

AccuVideo™imaging

YUV->RGB Index

21130

DAC
x3

R
G
B

256x24
VideoLUT

256x8x3
GraphicsLUT

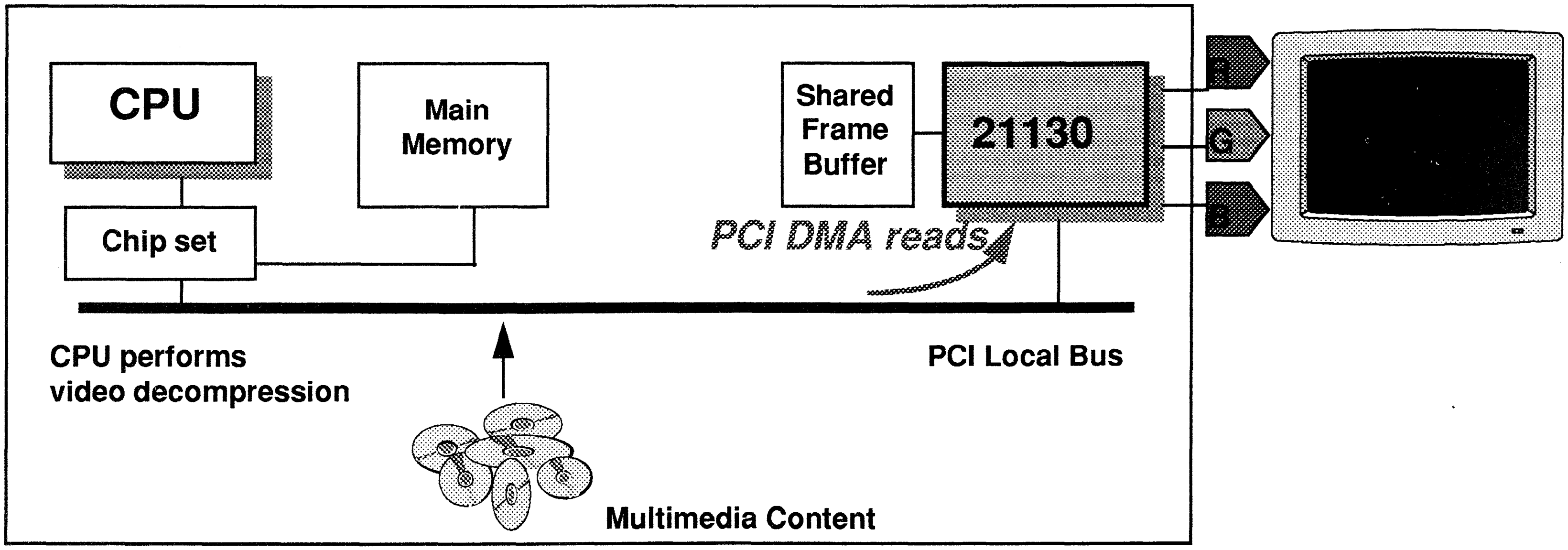
8-bit RGB index

DRAM Frame Buffer

Video
Stencil
1-bpp

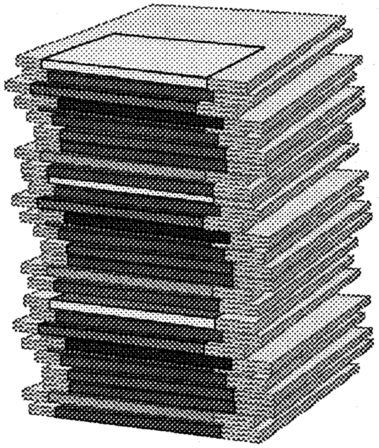
System Architecture Advantage

- Value segment is dominated by software codecs
  - Indeo and Cinepak
  - MPEG-1 emerging in late 95
- 21130 is a PCI Bus Master
  - Increases software codec performance by >20%



21130 Software Drivers

- Windows 3.1
- Windows95
- Windows NT 3.5 (Alpha and Intel)
- Video for Windows
  - DCI
- OS/2
  - ENDIVE
- SCO Unix
- AutoCAD



21130
Uncompromised Video Acceleration

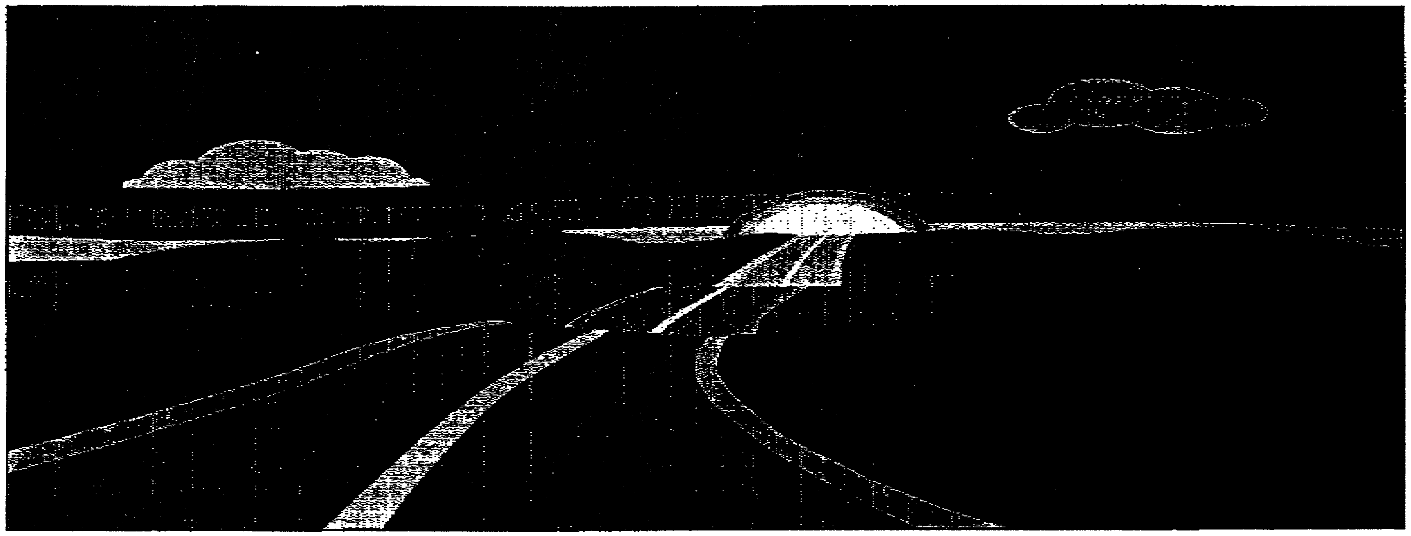
- Quality solution for mainstream business video and graphics
- Optimized video playback solution for SW codecs
- Flexible video windows with AccuVideo
- Balanced graphics and video performance

Driving Toward a PCI-Centric Graphics/Video Solution

Ken Lowe
Sierra Multimedia
2075 N. Capitol Ave.
San Jose, CA 95132
408-263-9300

Sierra Semiconductor

Driving Toward a PCI-Centric Graphics/Video Solution



Ken Lowe

Director of Marketing

Sierra Multimedia

Sierra Semiconductor

Relevant Market Trends

- Higher performance PCs
- Cross-platform adoption of PCI
- 64-bit GUI accelerators
- Integrated video acceleration
- Soft CODECs
- 3D acceleration support

While maintaining

- A constant-price market model

Graphics/Video Complex Interfacing Needs

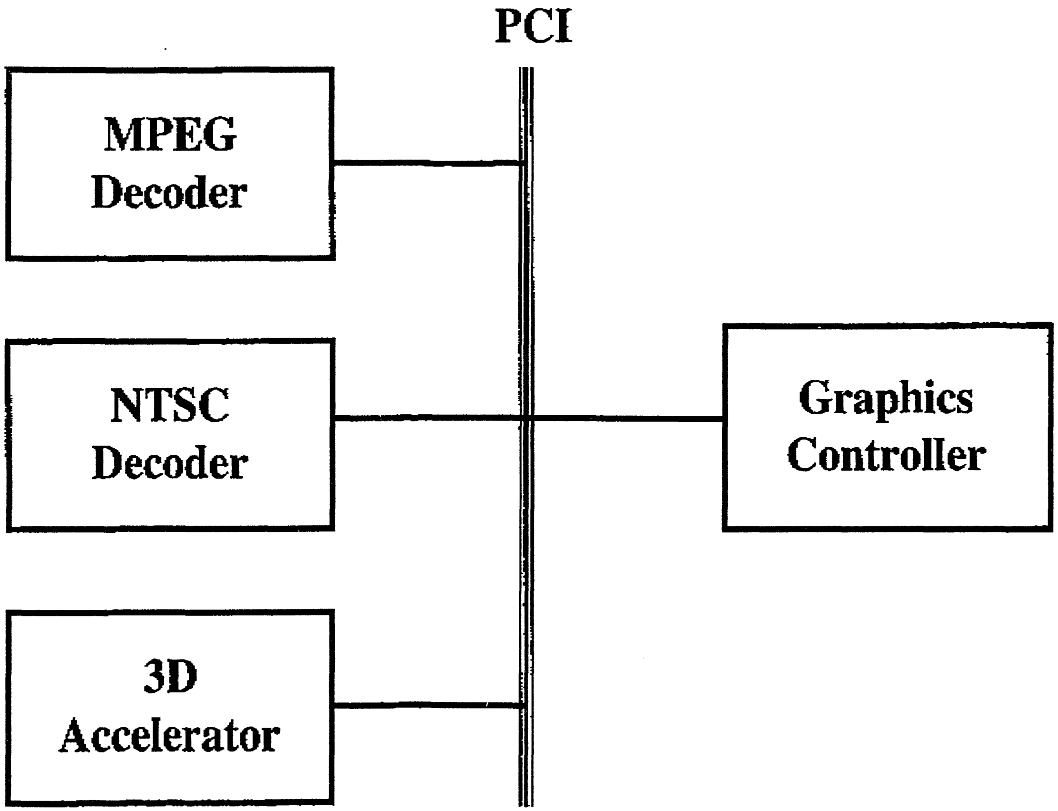
- Standard method of connection
  - Graphics controller
  - MPEG decoders
  - NTSC decoders
  - 3D Accelerators
- Low-cost
- Low pin-count

Alternatives and Issues

- VAFC - only supports video back-end mixing
- Philips - only supports uncompressed video inputs
- VMC - low acceptance, high cost (gates, pins)
- Custom - becomes “vendor-centric” = risk

Sierra Semiconductor

The PCI-Centric Approach



PCI

MPEG Decoder

NTSC Decoder

3D Accelerator

Graphics Controller

The PCI-Centric Benefits

- Single existing standard
- Inherently plug-n-play
- Lower cost/pins
- All data available to CPU

Sierra Semiconductor

Conclusions

The PCI-Centric Approach Provides

- Users with a universal, upgradable, plug-n-play environment
- PC vendors with a chip-vendor independent architecture
- Device vendors with a low-cost/low pin-count interface

Improving PCI Connectivity

Mercury Computer Systems, Inc.

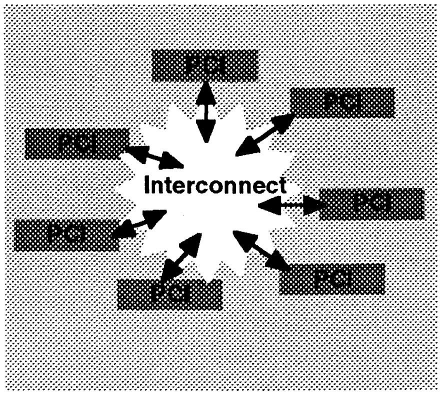
Barry S. Isenstein
Mercury Computer Systems, Inc.
199 Riverneck Road
Chelmsford, MA 01824
(508) 256-1300 and FAX (508) 256-3599

Improving PCI Connectivity

- Connectivity goals
- Physical and logical limitations
- Solution
- Case study

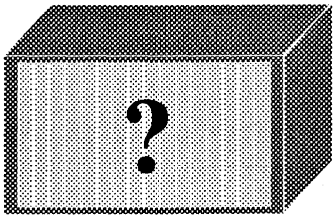
The Goals

- Connect dozens of high-performance PCI devices seamlessly
- Balance multiprocessing computation throughput with PCI I/O bandwidth
- Minimize latencies



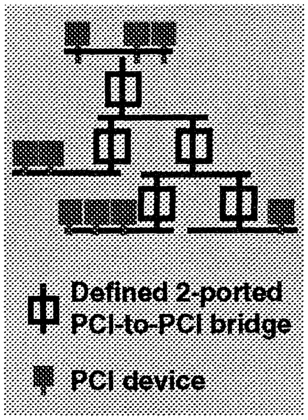
PCI Physical Limitations

- Connectivity of a single PCI bus limited by
  - 10 loads
  - 4 plug-in boards
- For multiprocessor systems, connectivity must extend beyond these limits
- Yet it is desirable to use PCI as an I/O bus



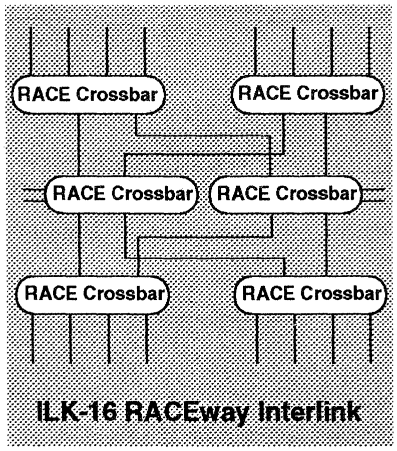
PCI Logical Limitations

- The hierarchical structure of two-port bridges
- Contention problems
- Latency problems
- Limited bisection bandwidth



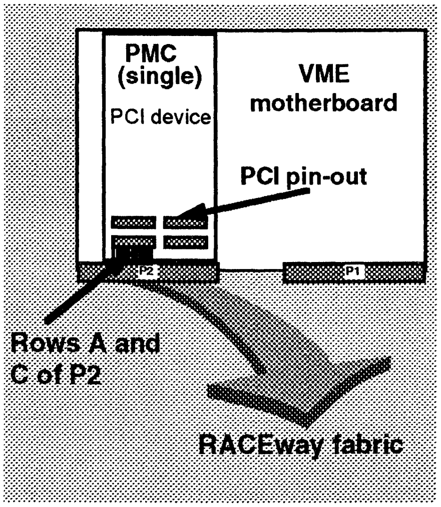
The Solution
RACEway Interlink Switching Fabric

- A switching fabric for PCI
- Implemented with glueless building blocks for scalable flexible interconnect
- Concurrent transactions each at PCI bandwidth
- Industry standard connection and protocol
- Requires a RACEway bridge chip for PCI



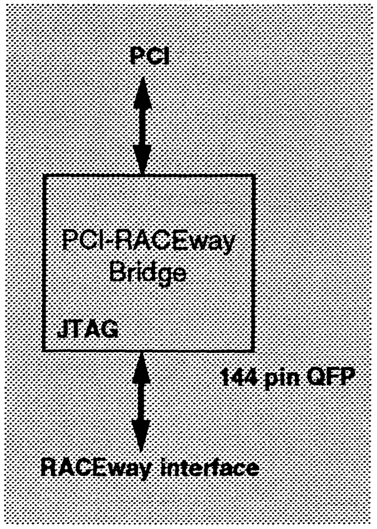
PMC Standard

- PCI Mezzanine Card (PMC) defines daughtercards with a PCI interface for VMEbus, Multibus, and Futurebus+
- A RACEway to PCI bridge on PMC daughtercard allows any PMC-equipped product to attach to the RACEway



PCI-RACEway Bridge ASIC

- 32-bit, 33 MHz PCI interface
- 32-bit, 40 MHz RACEway
- Performance
  - 132MB/s peak bandwidth
  - Up to 100MB/s sustained on 256 byte blocks
- Other support
  - DMA controller
  - Realtime clocks
  - Performance metering
  - Address mapping

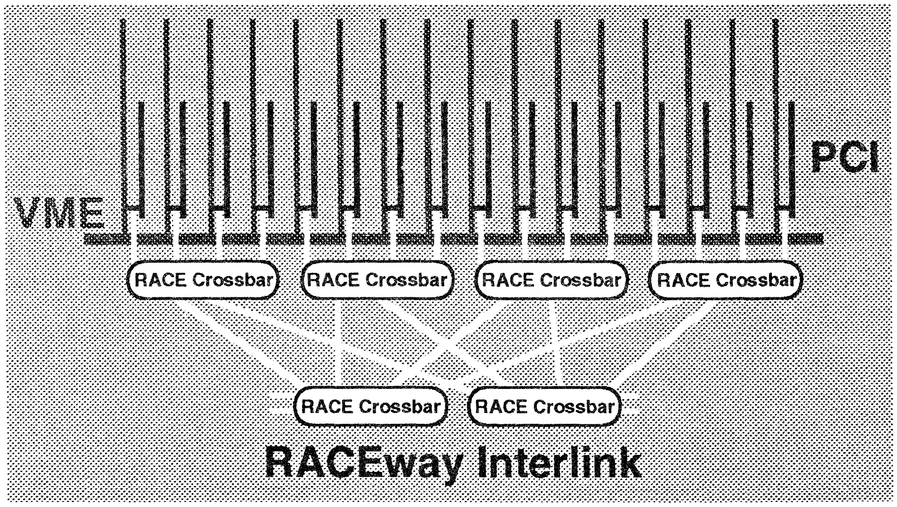


A Case Study

- Scalable PMC chassis with up to 16 slots and 32 PMC cards
- Applicable to VMEbus chassis and other PMC platforms
- 1.2 Gbyte/s peak aggregate bandwidth
- 640 Mbytes/sec peak bisection bandwidth
- Under three microsecond write latency
- Six component interconnect, at one Watt and one square inch each
- Performance metrics

Case Study Configuration

Standard VMEbus Chassis



Summary

- Two-ported PCI bridges limit configurability and performance for PCI bus interconnectivity.
- Existing standards (VME, PMC, and RACEway Interlink) provide a basis for system-wide PCI connectivity.
- PCI-RACEway bridge ASIC developed at Mercury is aimed at high-performance systems that balance multiprocessing computation with PCI I/O bandwidth.

Computer Systems, Inc.
MERCURY

PCI - CACHING SCSI HOST ADAPTER

Kamal Mansharamani
DCM Data Systems,
Vikrant Tower, 4, Rajendra Place,
New Delhi-110008, INDIA

ABSTRACT

This paper goes into the technology issues involved in the design of a high performance PCI caching SCSI host adapter. It discusses the design issues involved to deliver high throughput on servers which are based on the high performance CPU's like Pentium, DEC Alpha and POWER-PC. These CPU's can handle large amount of data which cannot be provided by the hitherto existing bus architectures. This problem has been addressed by the PCI bus. Even though the PCI bus can handle data transfer rates of upto 132 Mb/sec, the storage medias (which normally are SCSI) can deliver data only upto a maximum of 20 Mb/sec. In a server environment, the data transfer to/from the storage media is the primary bottleneck. In a conventional controller, this leads to an inefficient bus utilisation. This problem can be circumvented by having a Caching host adapter with an onboard controller with intelligent software to relieve the host from Input/Output tasks.

INTRODUCTION

In the recent past, technological advances have dramatically pushed up the CPU performance. This has been made possible on account of architectural enhancements as well as higher clock speeds. Almost all the new generation CPU's are 64 bit wide with super scalar architecture with multiple integer and floating point units allowing multiple instructions per clock. These chips also come with local instruction and data cache. In addition, the clock speeds have really gone up from 33 Mhz to more than 100 Mhz. In fact the DEC ALPHA chip operates at 250 Mhz and above. With the result, there has been a multifold increase in the CPU performance.

Almost all of these CPU's have an extremely fast bus interface and need a bus bandwidth in excess of 100 Mb per second for optimum performance. However, none of the existing buses like EISA or the Microchannel have the required bandwidth to cater to the high performance CPU's. The advent of PCI bus has eliminated this bottleneck. The PCI bus supports a bandwidth of 132 Mb/sec and can match upto the speed requirements of a high performance CPU.

DESIGN ISSUES

The design of a SCSI host adapter sub system to cater to this environment of high performance CPU's and high bandwidth PCI bus poses new challenges. The key design issues for a SCSI host adapter with possible solutions have been discussed in this paper.

1) Bus Utilisation

Since PCI is a high performance local bus, its optimum utilisation is of paramount importance. An efficient use of the bus is essential to enable all the bus masters to take advantage of its high bandwidth. What this essentially means is that the bus master should be able to get 'on' the PCI bus quickly and also get 'off' the bus as soon as the transfer is over. Also, while the master is on the bus, it should be able to transfer data at the full PCI speed. Latency is a parameter which controls the bus utilisation. A designer has to ensure that the adapter does not hog up the bus.

DCM Data Systems has designed a PCI SCSI host adapter which could make hundred percent utilisation of the PCI bus. For this a FIFO of 64 DWORD depth has been used at the PCI end, to ensure that the data transfer on the PCI bus happens on every clock edge. This would ensure that the host adapter would occupy the bus only when it is required. Also an innovative scheme has been used to implement the FIFO.

The FIFO has been implemented as a dynamic circular buffer in which the read and write operations can take place independently at different rates. The FIFO is managed through two flags which are called FIFO_Empty and FIFO_Full. In addition, two more flags have been provided to fine tune the performance, especially taking into account the PCI latency given to the card and the access speed of the DRAM cache. These flags are referred to as READ-ONLY and WRITE-READY flags. These flags are dependent on the threshold values which can be programmed through registers. The READ threshold is used to specify the minimum amount of data in the FIFO before a READ operation is triggered off. Similarly, the WRITE threshold is used to specify the minimum threshold in the FIFO before the write operation can begin. If the read and write operations are correctly programmed, then the FIFO can give very high sustained bursts. For example, if the READ and WRITE rates are the same, an infinite burst can be theoretically supported. The maximum burst which can be supported through the FIFO can be given by the following equation :

$$\text{Burst size} = 64 * \sum_{i=0}^{\infty} \left(\frac{x}{y}\right)^{i}$$

x = No. of clocks required to read/write the FIFO buffer from the PCI Bus

y = No. of clocks required to read/write the FIFO buffer from the DRAM.

This is on the assumption that DRAM access are slower than the PCI access. The SCSI host adapter designed by DCM Data Systems supports upto 32 MB of DRAM cache. It takes 3 clock cycles to write one DWORD from the DRAM into the FIFO, while it takes two clock cycles to transfer one DWORD from the FIFO onto the PCI bus. The read and write operations work concurrently on the bus. Let us take the case where the FIFO is full and we initiate a transfer on the PCI bus. In this case the controller would keep pumping the data from the FIFO onto the PCI bus, while at the same time it would keep filling the FIFO by taking the data from the DRAM. In this scenario, a maximum number of 186 DWORDS could be transfered. So a design implementing this scheme can transfer a full block of data (512 Bytes) at full PCI speed of 132 MByte/second.

2) Caching

The PCI bus has the capability to match the demands of high performance CPU's in terms of bus bandwidth and one can design a controller which can operate at the peak PCI speed of 132MB/sec. However, the storage media still acts as a bottleneck. The fastest data transfer rate one can obtain from a SCSI storage system is only 20MB/sec.

Caching host adapters play an important role in bridging the gap between the I/O requirements of the high performance CPU's and the slow storage media. Even through the new generation operating system like WINDOWS-NT, Novell Netware have some form of cache built in, the hardware cache on the host adapter can greatly enhance the performance of the system. The caching logic built in the Operating System is not tuned for a particular type of host adapter - hard drive combination, but is a general implementation, whereas, by having caching logic on the host adapter, one could tune it to the host adapter-hard drive combination to give the best results under all situations.

The caching logic implementation consists of two main strategies :

(a) Intelligent Read Ahead Strategy

When the data is being accessed in a sequential manner, then it is beneficial if, by giving one command to the hard drive, the entire track is read off. The data that has been "read-ahead" would be retained in the cache, and for any subsequent read call, the data would be returned from here and there would be no need to give the command to access the hard drive. For example, for a hard drive with 32 sectors per track, the time taken to read one track would be :

With no read ahead

32 *(time taken to make one command+
time taken to read one sector+
time taken to transfer data for one sector+
Interrupt processing time)

With read ahead

(Time taken to make one command+
Interrupt processing time+
32 * (time taken to read one sector+
time taken to transfer data for one sector))

Clearly, the saving of time is obvious.

However, the amount of memory on the host adapter is limited, and all the tracks which are "read-ahead", have to be maintained in this limited memory. It would not be advisable to read ahead data (which may not be used later) and store in the memory, at the expense of throwing out data which is being used more often. Because, that would defeat the whole purpose of caching which was to minimise the traffic to the hard-drive. To decide, as to when to do the read-ahead and when not to do, the following strategy can be followed.

If "N" sectors are read from a track sequentially, then the read for the (N+1) sector would result in the entire track being read. The "N" is a variable parameter and is tunable. Typically, "N" could be 20% of the number of sectors per track. For example, for hard drive of 32 sectors per track, the value of "N" could be 6. So if sectors 1, 2, 3, 4, 5 and 6 of a track are read sequentially, then a read of sector number 7 of this track would result in all the sectors of this track (i.e. 7 thru 32) being read.

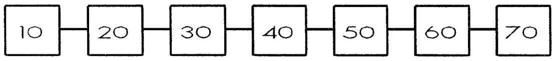
This feature is very useful in SERVER environment, when the database is being created, or when the database is being scanned for some information sequentially or when the database has to be backed up.

(b) Delyed Write Strategy

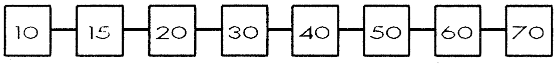
In DELAYED WRITE STRATEGY, any write call that the Operating System may issue for writing the data, would be delayed to the extent possible and the actual write to the hard drive would be committed much later. This ensures that if later the same data has to be modified or over written, then the update would take place in memory and the overhead of preparing the command and issuing the command to the hard drive would be avoided. In a SERVER environment, the updates to the same data block keep happening and by delaying the writes, the performance improvement is tremendous.

Further improvements can be achieved by implementing scatter-gather and elevator sorting. In scatter-gather, one tries to combine as much of read or write commands as possible so that the number of commands that are issued to the device are minimised.

Results have shown that seek time and latency time of the hard drive are a significant portion of the total time taken to complete one I/O request. If one could minimise the seek time, an improvement in performance would be seen. The commands that are issued to the drive are issued in such a fashion, that the head keeps moving in one direction. Care has to be taken that no command is starved because of elevator sorting, i.e. each command should finish within a pre-determined time and that no command is waiting indefinitely. An out-of-order count can be maintained to know the number of commands that have come which are out-of-order. For example, if the I/O requests that have come are for, lets say sectors in the following order:-



and we are processing I/O request for sector 30 (the third I/O request) and another request comes for sector 15, then the request chain would look like



and the out-of-order count would be one. Once this out-of-order count goes upto, lets say 5, then any further requests would be put in a seperate "batch" and all requests for this batch would be serviced first. Otherwise, the request for sector 15 could get starved as we are moving in the forward direction.

(c) Concurrent DMA Operation

To achieve a good throughput, the data transfer between the SCSI device and the PCI host is done through a caching mechanism, wherein the data is trasferred via the DRAM. A typical read from the SCSI device, for example, is broken up into various scatter gather dma commands, which are chained. Incase some of the blocks are found in the DRAM (cache) they can be transferred directly to the host, without a fetch. The entire operation of transfer is carried on by two DMA channels, which operate concurrently. One DMA transfers the data between the SCSI and the cache, where as the other transfers the data between the cache and the PCI host. As the DRAM transfer rate is not the same as the transfer rate of the SCSI device or the PCI, two independent FIFOs are used to buffer the data during the transfer. This allows a much better utilization of all the buses.

(d) The Multi Ported DRAM

The DRAM is used as the cache and is multi ported as it has to be used by all the masters. Each host that requires to access the DRAM is given a burst access. This is done to utilize the DRAM bus bandwidth very effectively. To achieve a burst in the case of the local CPU, we have incorporated independent read-ahead and write-back buffers of 16 bytes each. The read-ahead buffers prefetch a 16 byte block from the DRAM after the current burst is over. Subsequent accesses to DRAM within this block can be satisfied by this buffer. If data is not found in the buffer, the CPU will have to wait for the current burst to get over. The read buffers also snoop on all the DRAM cycles and update the data in case of writes in the range at which the data is in the buffers. For the write operation there are 16 bytes of write back buffers. The writes need not be contiguous but would have to be DWORD aligned for full utilisation of the buffers.

Once data is written onto a buffer, a flush request is raised to the DRAM arbiter, the CPU flush operation gets the highest priority for data consistency. This improves the throughput considerably on the DRAM bus, especially if the cpu accesses data within 16 byte boundary.

CONCLUSION

The intelligent SCSI host adapter, in which the Firmware is very tightly coupled with the hardware, allows a much finer control on the throughput. The key parameters - the latencies to various DRAM masters and the sizes of transfers to/from the host are programmable from the host driver. This allows the performance of the controller to be fine tuned according to the Operating System under which the controller operates. Caching has been evaluated under NetWare. It has been observed that without caching, the number of I/O requests waiting were about 2000, whereas with caching, the number of I/O requests waiting came down to 3-4, a phenomenol improvement in response time.

This shows that it is possible to build Intelligent SCSI host adapter with the current technology which can operate at the maximum PCI speeds. In addition an innovative caching algorithm with support from the hardware can significantly enhance the performance of the system specially under heavily loaded server environment.

REFERENCES

Edward, S and George Willse PCI Hardware and Software Architecture and Design. Annabooks, San Diego USA 1994.

Shanley, T and Don Anderson PCI System Architecture 1993, 1994 Mindshare, Inc. Texas, USA 1994.

THE PCI INTERFACE PROPELS DISK DRIVES TO MEET NEW BANDWIDTH REQUIREMENTS.

Danial Faizullabhoy
Adaptec Corp.
691 S. Milpitas Blvd.
Milpitas, CA 95035
Phone: (408-945-8600)

Abstract

Today, graphical user interfaces, multimedia applications, and the rising popularity of ATM and Fast Ethernet networks have mounted unrelenting performance pressures on both the PC host interface and the mass-storage subsystem. For host interface buses, the PCI bus is rapidly becoming the interface of choice for the modern PC. PCI supplies the performance needed by supplying bandwidths of 132 MB/sec. Additionally, the PCI roadmap go to 792 MB/sec. Other performance features in the PCI architecture include full multiple-master capability with arbitration. All of this propels the elements of mass storage technology to higher performance levels.

For mass storage device interfaces, new interfaces with much higher bandwidths are up to the task to support the bandwidth requirements for the PCI bus. Interfaces such as DoubleSpeed SCSI, Serial Storage Architecture and Fibre Channel Arbitrated Loop supply bandwidths from 40 MB/sec to 200 MB/sec. This is much higher than today's ISA based disk drive.

Another technology element driving the bandwidth equation is the rising NRZ data transfer rates available from hard disk controllers. Areal densities are rising 60% per year and disk rotation rates are at 7200 rpm. The resulting sustained bandwidths off of the disk have risen from 10 MB/sec in 1993 to over 20 MB/sec in 1995. This has forced hard disk designers to look for new, high bandwidth solutions in their controller designs.

The goal of the article is to examine the impact of the PCI interface on mass storage architectures. First, the bandwidth requirements of emerging applications will be briefly reviewed. Then overall technology trends in the disk drive market will be examined. Finally, a detailed discussion of a high bandwidth disk controller feature is presented

The discussion will focus on disk controller technologies used in Adaptec's new AIC-8300 products. These products have the automation and internal architectures that will effectively support PCI's high bandwidth requirements. Automation techniques used to increase bandwidth will be discussed. This includes automated data transfers, host automation and faster/flexible ECC. Other features used to fine tune hard disk controller performance are reviewed. The net result is a hard disk controller architecture that supports the requirements for PCI bus based application in 1995 disk drive designs.

NEW MASTER MODE ENHANCED IDE CONTROLLER

Tzu-Mu Lin
VIA Technologies, Inc.
5020 Brandin Court
Fremont, CA 94538 USA

ABSTRACT

VIA Technologies' new VT83C571 Master Mode Enhanced IDE Controller takes full advantage of PCI bus mastering. It approaches SCSI performance by offering Enhanced IDE Mode 4 data transfer rates of 16.6MB/s, with PCI burst transfers of up to 132MB/s.

VIA's third generation technology is the first IDE master mode controller to combine PCI SIG compatibility with a 128 byte FIFO cache, the largest to date.

The new 100-pin PQFP packaged VT83C571 chip is a bridge between the PCI bus and IDE devices. It complies with both the latest Enhanced IDE and PCI Rev. 2.0 specifications.

Its key advantages are

- Efficient PCI bus utilization
- Concurrent DMA channels
- Flexible byte alignment
- Flexible FIFO assignment

EFFICIENT PCI BUS UTILIZATION

The 571 reduces to a minimum the time that it uses the PCI bus, thereby enabling bus resources to be allocated to other PCI bus masters. It achieves this goal through the use of intelligent scatter-and gather (S&G) protocol and its ability to control data flow

The S&G protocol is based on a PCI-SIG proposal which in the future will be incorporated in most operating systems. After issuing only one command and pointing to the S&G list, the host CPU hands the transfer task over to the controller. Thereafter the 571's command processor manages block data transfers without host intervention letting the host CPU perform other tasks while IDE operations run in the background.

The data flow control manages the load of traffic on the PCI bus while it performs IDE transfers into its internal FIFOs. The 571 always tries to accomplish doubleword transfers on the PCI bus PCI bus utilization for a sector (512 byte) transfer is approximatly as follows:

- Arbitration_overhead +
- (Read Descriptor) +
- 128 * PCI-cycle + Two

CONCURRENT DMA CHANNELS

Two DMA engines are built-in which enables each channel to operate independently and concurrently. When two channels simultaneously request DMA services from the DMA engines, the arbiter automatically switches between the two channels to maximally utilize the IDE data bus.

FLEXIBLE BYTE ALIGNMENT

Data is transferred to and from peripheral devices and system memory either as a word (2 bytes) or doubleword (4 bytes). In those cases where the base address is not doubleword aligned or the byte count is not doubleword aligned, the transfer on the PCI bus will still be accomplished in doublewords and burst transfers will be maintained on the PCI bus. This feature also eliminates double-buffering by the driver which in turn improves system performance.

FLEXIBLE FIFO ASSIGNMENT

The 571 supports 32 level doubleword FIFO as prefetch and post write buffers for the two IDE channels. The 32 level FIFO can be shared between the two channels under programming control. One of five configurations is available: 32/0, 24/8, 16/16, 8/24 and 0/32. In this manner the system can be optimized for different peripheral speeds by flexibly assigning FIFO between the two channels. When only one channel is used, which is usually the case, the depth of the FIFO is effectively double that of hardwired FIFO configurations.

Hard Disk Technology

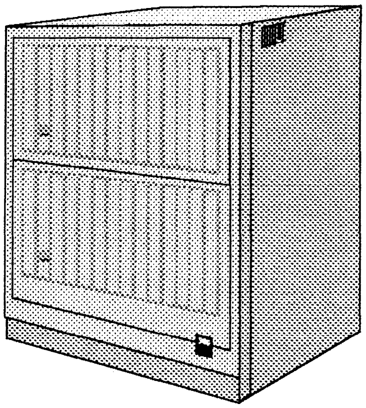
Donald F. Coffin
Future Domain Corporation
2801 McGaw Ave.
Irvine, CA 92714
714-253-0508

Hard Disk Technology

Donald F. Coffin
Product Marketing Manager
Future Domain Corporation

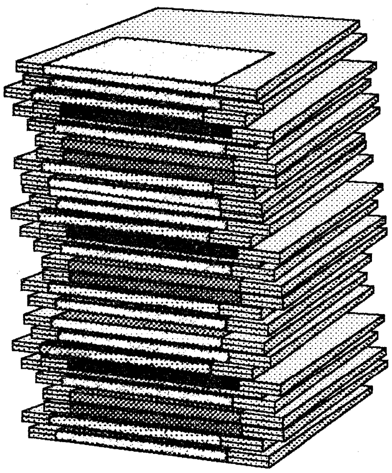
Hardware Interfaces

- RLL
- ESDI
- MFM
- ST-506
- IDE (DMA)
- SCSI



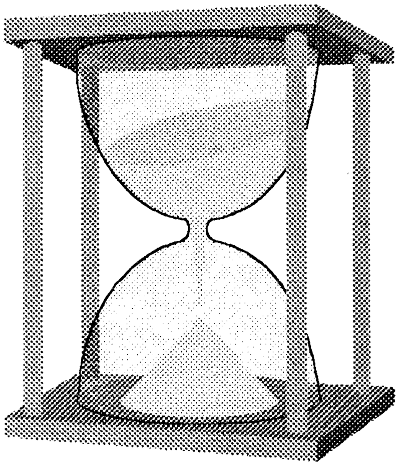
Sizes

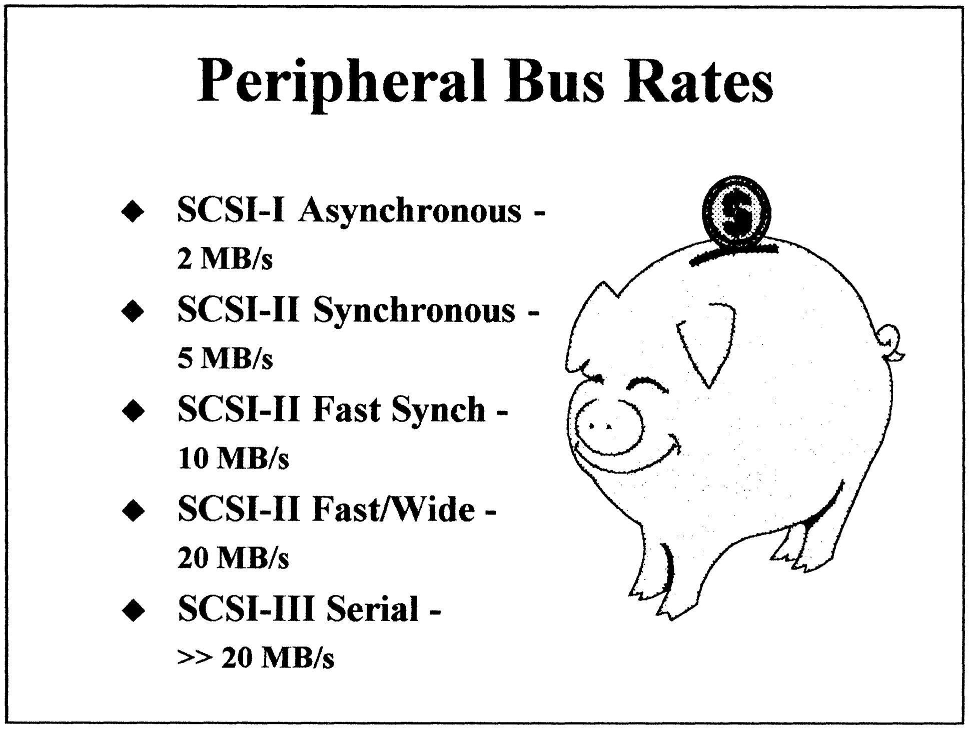
- < 32 MB (DOS limitation)
- < 528 MB (CHS limitation)
- < 2 GB (SCSI-I 6-byte CDBs)
- < 8 GB (SCSI-II 10-byte CDBs, EIDE)
- > 8 GB



Data Rates

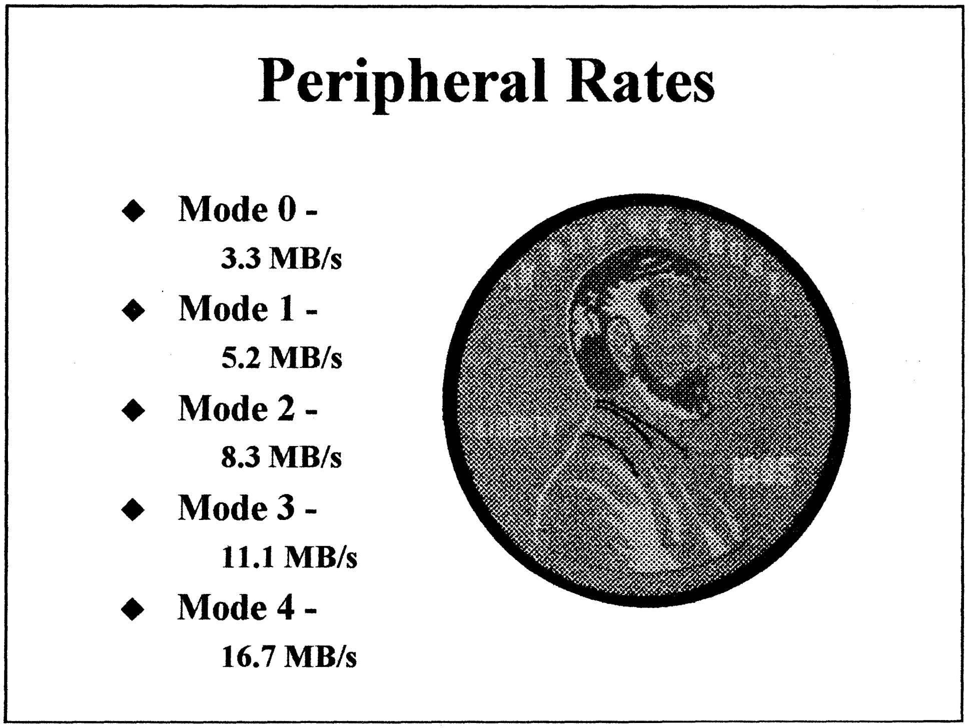
- Past:
  - Peripheral thru-put limited by host bus
- Future:
  - Realized thru-put limited by peripheral





Peripheral Bus Rates

- SCSI-I Asynchronous - 2 MB/s
- SCSI-II Synchronous - 5 MB/s
- SCSI-II Fast Synch - 10 MB/s
- SCSI-II Fast/Wide - 20 MB/s
- SCSI-III Serial - >> 20 MB/s



Peripheral Rates

- Mode 0 - 3.3 MB/s
- Mode 1 - 5.2 MB/s
- Mode 2 - 8.3 MB/s
- Mode 3 - 11.1 MB/s
- Mode 4 - 16.7 MB/s

Host Bus Rates

- **ISA**

  8 MHz

  × **16 data bits (2 bytes)** → = **4-8 MB/s**

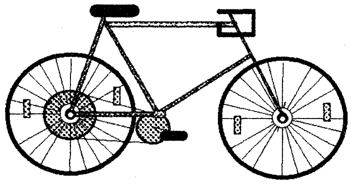
  ÷ **2-4 clocks/cycle**

- **EISA**

  8 MHz

  × **32 data bits (4 bytes)** →

  ÷ **1 clock/cycle** = **32 MB/s**



Host Bus Rates

- **MCA**

  10-20 MHz

  × **32 data bits (4 bytes)** → = **40-80 MB/s**

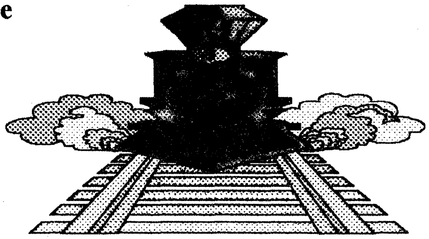
  ÷ **1 clock/cycle**

- **VL**

  33 MHz

  × **32 data bits (4 bytes)** → = **132 MB/s**

  ÷ **1 clock/cycle**



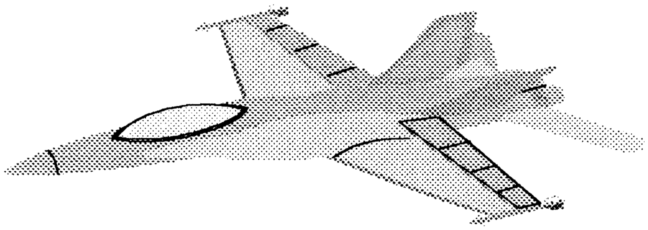
Host Bus Rates

- **PCI**

  33 MHz

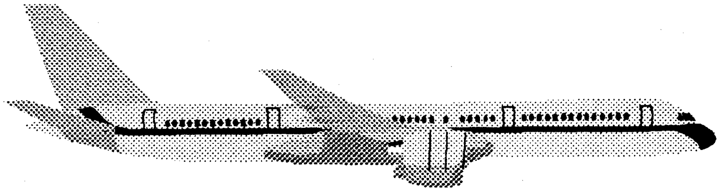
  × **32 data bits (4 bytes)** ➡ = **132 MB/s**

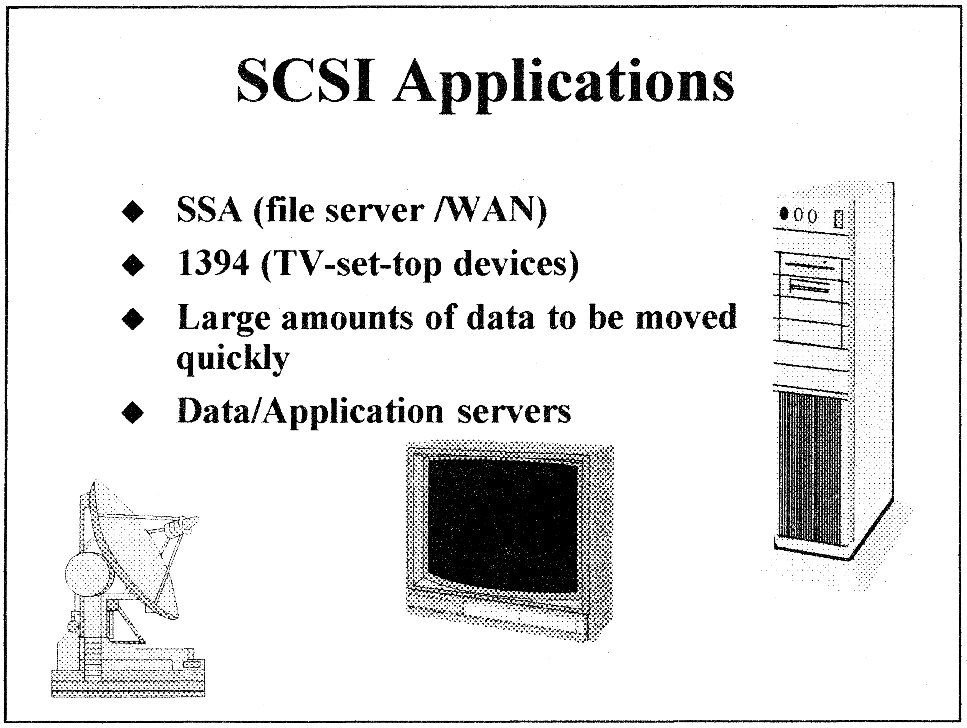
  ÷ **1 clock/cycle**



Bottom Line

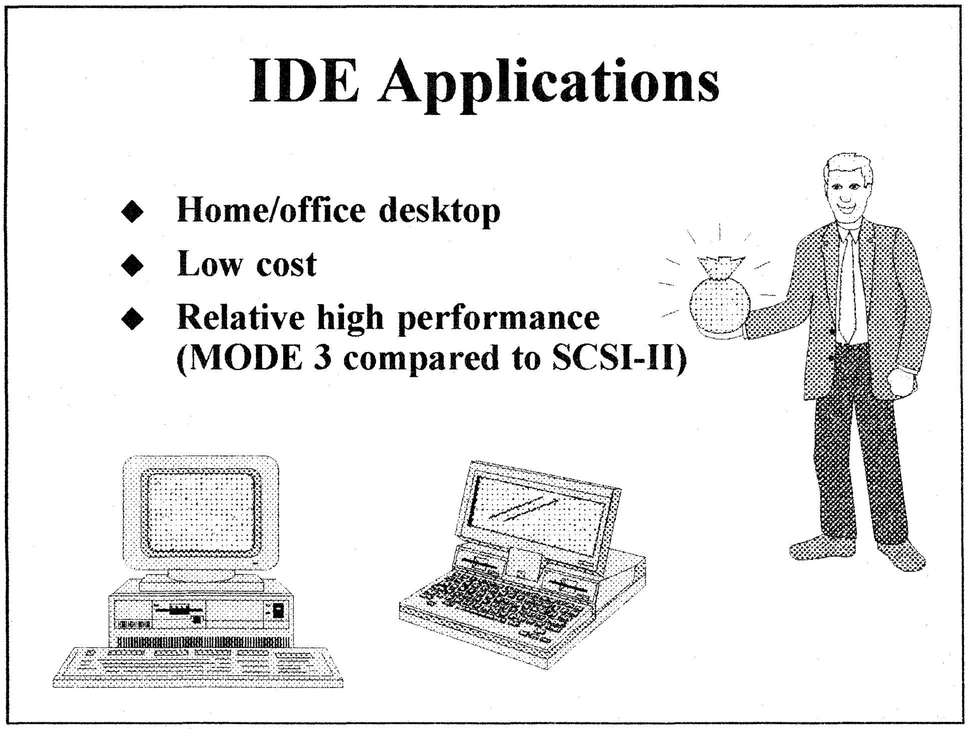
- **Greater thru-put**
- **Processor independent**
- **More bus for your processor**
- **Data flies to and from disk.**





SCSI Applications

- SSA (file server /WAN)
- 1394 (TV-set-top devices)
- Large amounts of data to be moved quickly
- Data/Application servers



IDE Applications

- Home/office desktop
- Low cost
- Relative high performance (MODE 3 compared to SCSI-II)

PCI-SCSI Made Simple

Peter Passaretti
Vice-President, Marketing and Sales
Initio Corporation
2901 Tasman Dr., Suite 201
Santa Clara, CA 95054
(408) 988-1919/3254 (fax)
peterp@initio.com

The language of PCI and SCSI buses, if not the concept, is foreign to most of us. This presentation will try to explain the need for using the SCSI bus together with the PCI bus. Via simple visuals, the viewer will gain an understanding of the architecture of the PCI-SCSI controller. The presenter will take the viewer through the current construction trends, ending with a prediction of future SCSI adapter directions and implementations.

Theoretical Benchmarking

By: Yu-Ping Cheng

AdvanSys
1150 Ringwood Ct.
San Jose, CA 95131
Phone: (408)383-9400
Fax: (408)383-9612

Abstract

Everyone wants to know how well an adapter works, therefore benchmark programs for I/O device adapters became necessary. Typical benchmark programs measure the data throughput. Unfortunately, the throughput depends on many things outside an adapter such as: the computer and I/O device speed, the system bus bandwidth, the amount of memory or cache available and, most importantly, the overhead of the benchmark itself. It is very easy for a manufacturer to pick the unique combination to give his adapter the best results.

The benchmark number are accurate only from one specific perspective -- or the benchmark's perspective. This gives an overview of a disk adapter's: performance relative to these three parameters:

1. The adapter's overhead in processing an I/O request.
2. The adapter's data transfer speed.
3. The adapter's ability to do multitasking.

Hard Disk Adapter Performance

A typical disk drive operation consists of five major components:

1. Operating system software and device driver overhead -- x1.
2. Disk adapter overhead -- x2.
3. Disk drive command decode, seek, and rotational delays -- x3.
4. Disk adapter data transfer speed -- x4.
5. Amount of data transferred per I/O request -- x5.

The amount of throughput of a disk adapter is simply:

(time_under_test / (x1+x2+x3+(x5/x4))) * x5

The term (x1+x2+x3+(x5/x4)) determines the processing time of one disk I/O request. Dividing it by **time_under_test** yields the number of requests during the tests. Finally, the multiplication of x5 gives us the total data throughput. The total throughput will be limited by the system and peripheral busses. For example, one can not get more than 6 MB/sec on an ISA bus, 32 MB/sec on an EISA bus, and 132 MB/sec on a local bus. With only one SCSI bus, the maximum throughput is limited at 10 MB/sec. This formula is a very simple view of disk drive throughput.

An ideal benchmark should have **x1** close to zero. A large x1 gives a very low I/O throughput and a very distorted result. **x2** is provided by the adapter manufacturer. **x3** depends on the type of disk drives being used in benchmark and also depends on how the adapter can overlap multiple disk requests as we shall explain it in details later. **x4**, like x2, is provided by adapter manufacturer. **x5** is chosen by one who runs the benchmark. Typically, it relates to a specific application.

Multitasking Disk Adapters

High-performance adapters are designed to manage multiple requests in order to maximize throughput. There are few benchmark programs actually measuring the multitasking ability of an adapter. There are two ways of improving a disk adapter's throughput using multitasking:

1. Multiple commands to SCSI 2 disk drives.
2. Multiple disk drivers for overlapping disk seeks and rotational delays.

Supporting ISA Legacy Peripherals on PCI

Bert McComas
Chief Technical Officer
In-Stat, Inc.
7418 East Helm Drive
Scottsdale, AZ 85260
Ph. (602) 483-4448 Fax (602) 483-0400

As long as MS-DOS survives, applications running under it that interact directly with peripheral hardware will require ISA style support of DMA and interrupts. Though PCI provides much greater bandwidth than ISA, it will remain with us as a legacy for some time to come.

The need for DOS support is satisfied in the desktop market today by the inclusion of both the PCI and the ISA buses within systems. However, some special cases cannot be satisfied within the typical desktop configuration and demand a new solution. They include:

- PCI docking involving ISA buses in both the notebook and the docking bay

- PCMCIA over PCI requiring both ISA interrupts and DMA capabilities

- On-board PCI-based super I/O chips with floppy and or error-correcting protocols

- Compatible audio chips using the PCI interface

- Future systems lacking an ISA backplane

Each chipset or system vendor considering any of these types of systems or features must develop a proprietary method to circumvent the problems. Most have chosen variations of a near term brute force solution using many sideband signals. This allows near equivalent ISA functionality in every part of the PCI system. Though it can meet time to market requirements, this approach is not a worthy solution to a problem which will be with us for some time to come.

This is why industry leaders have banded together to establish an intelligent control protocol that will translate DMA activity from several ISA-like sources into PCI cycles, and allow ISA interrupt information to be transmitted using a single pin. This allows full ISA legacy support in notebook computers and their docking stations, PCI super I/O, and PCI audio chips, with the addition of one rather than many pins. It will also help consolidate the industry, which currently is saddled with several proprietary, high pin count solutions to the ISA legacy problem.

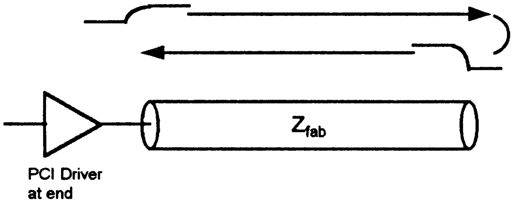
PCI Electrical Design and Bus Layout

Jim Murashige
Adaptec
691 S. Milpitas Blvd.
Milpitas, CA 95025
408- 957-4813

PCI Bus Layout Goals and Strategies

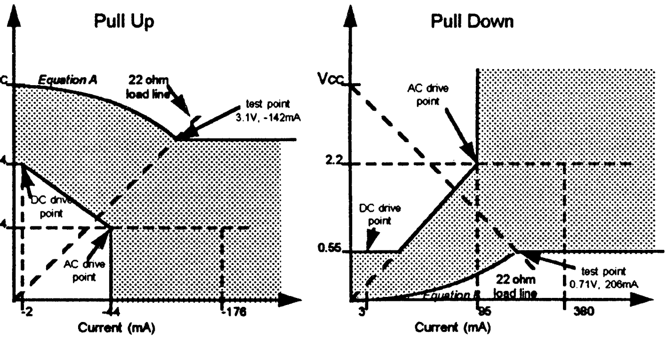
- Keep Loaded Characteristic Impedance **Up**(>32ohms)
- Meet PCI T_{PROP} Requirements By Either:
  Keeping Signal Velocity Up
  Shortening Bus Lengths
  T_{PROP} = Length/Velocity
- Minimize Impedance Discontinuities

PCI = Reflected Wave Switching



- Ends of the bus are UNTERMINATED
- A REFLECTION is Required
- Switching Threshold is Assured Only on Return Wave
- Devices on END, Switch First
- Worst Case Propagation Delay is One Round Trip(T_{PROP} **<=10ns)**

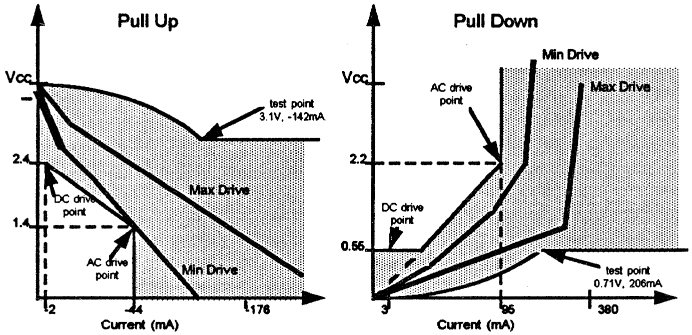
The AC Environment is KEY



V/I Curves for PCI 5V Signaling

- PCI Spec Defines AC Device Characteristics
- Characteristics are Different for Pull-Up, Pull-Down, 5V, 3.3V Signaling
- AC Characteristics Dictate the Required PC Board Transmission Environment

Commercial Devices Will Adhere to Min/Max Drive Requirements

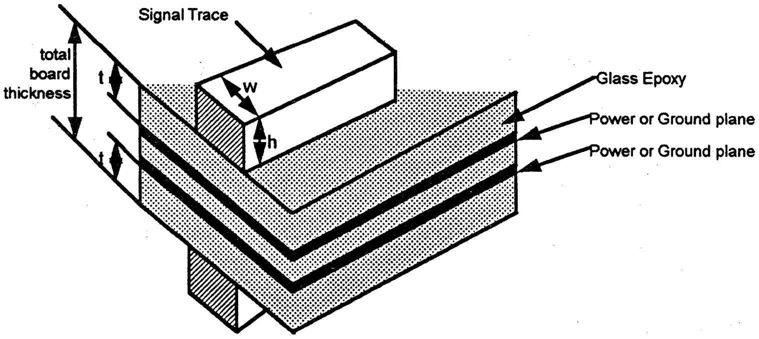


Example Adaptec AIC-7870

- PCI Defined AC Specs in Line With Current IC Technology
- Slew Rate Requirements Must Also Be Considered (1V/ns - 5V/ns)

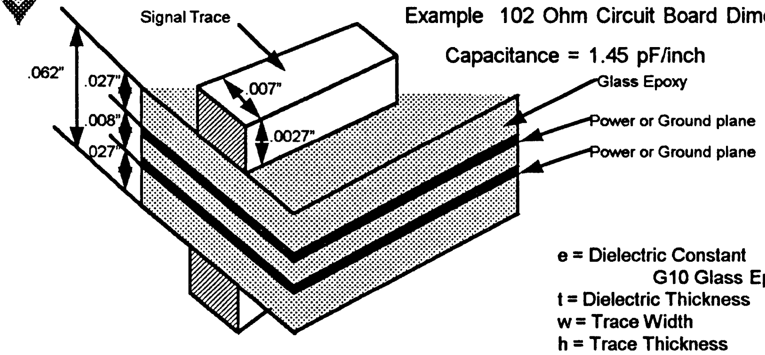
The Circuit Board is a Design Component

Micro Stripline Geometry



C - trace capacitance/inch is proportional to w/t
L - trace inductance/inch is proportional to 1/(w+h)
Characteristic Impedance/inch is $\sqrt{L/C}$
Propagation Delay/inch is $\sqrt{L*C}$

Calculation of Circuit Board Characteristics



Example 102 Ohm Circuit Board Dimensions

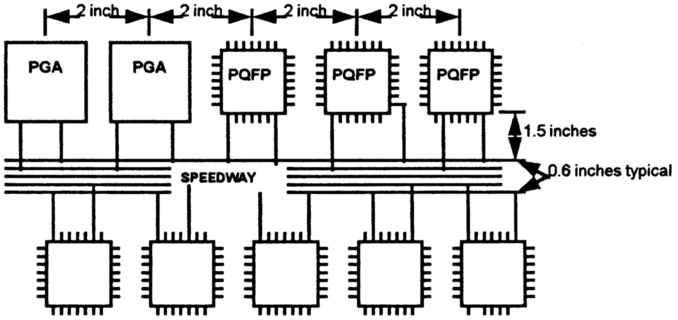
Capacitance = 1.45 pF/inch

e = Dielectric Constant
G10 Glass Epoxy=5
t = Dielectric Thickness
w = Trace Width
h = Trace Thickness

$$Z = \frac{87}{\sqrt{e + 1.41}} \ln\left(\frac{5.98t}{0.8w + h}\right) \qquad t_{pd} = 0.08475\sqrt{0.475e + 0.67} \text{ ns/in}$$

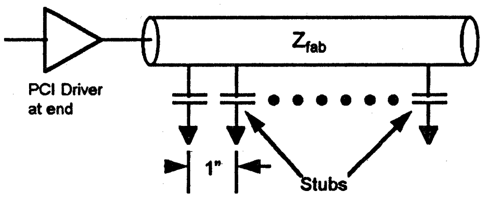
0.148 ns/inch for G10

PCI Speedway Layout



- A Suggested Layout From the PCI SIG
- A Good Example- Easy to Model
- Demonstrates Signal Transmission Principles

PCI Speedway Transmission Model



- Devices Are Evenly Spaced
- Loading Effects Are Principally Capacitive
- Short Stubs Minimize Stray Capacitive Loading
- ICs Are Limited to 10pf Loading

Loaded Impedance > 32 ohms
T_{PROP} < 10ns

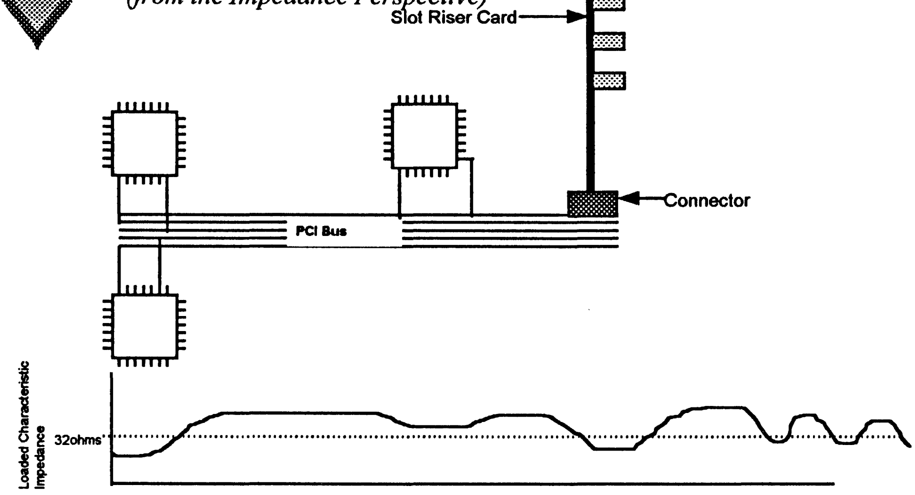
The Debilitative Effects of Capacitance

- **Capacitance Slows Down Signals, Lowers Effective Impedance**

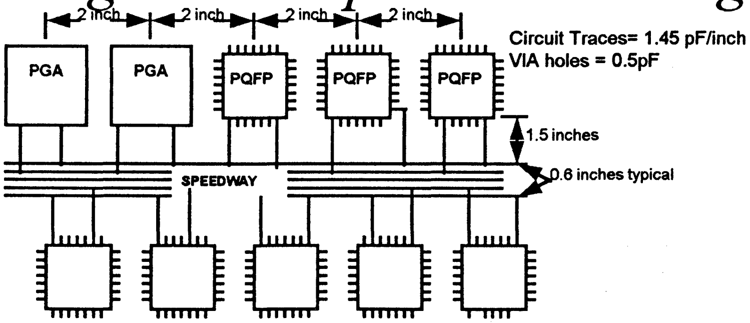
- **Evenly Distributed Capacitance is Better than Lumped Capacitance**
  Easier to Model
  Lumped Capacitance creates impedance discontinuities and more noise, in effect - a PCI “Speedbump”

A Not So Good Bus Layout

(from the Impedance Perspective)



Factoring In the Capacitive Loading



$$C_{load} = C_{via} + C_{stub} + C_{via} + C_{i/o}$$

$$C_{load} = 0.5pF + (1.45pF/in * 1.5in) + 0.5pF + 10pF$$

$$C_{load} = 13.175pF$$

$$\text{Adjustment factor} = \sqrt{\frac{(1.45 + C_{load})}{1.45}} = 3.176$$

Loaded Impedance and Propagation Delay

Loaded Characteristic Impedance = Characteristic Impedance/Adjustment Factor

Loaded Propagation Delay = Propagation Delay* Adjustment factor

Loaded Impedance = 102 ohms / 3.176 = 32.1 ohms

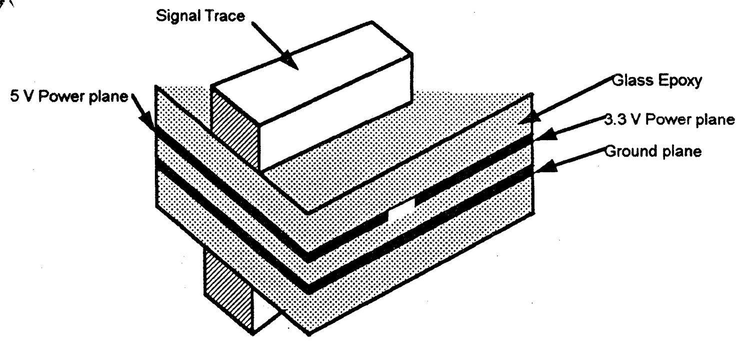
Loaded Prop Delay = 0.148 ns/in * 3.176 = 0.47ns/in

Round-trip Trace Length = T_{PROP} / Loaded Prop Delay
= 10ns / 0.47ns/in = 21.28 inches

One Way trace Length = 21.28 inches/2 = 10.65 inches

Note: Narrower Traces = Less Capacitance = Better Signals!

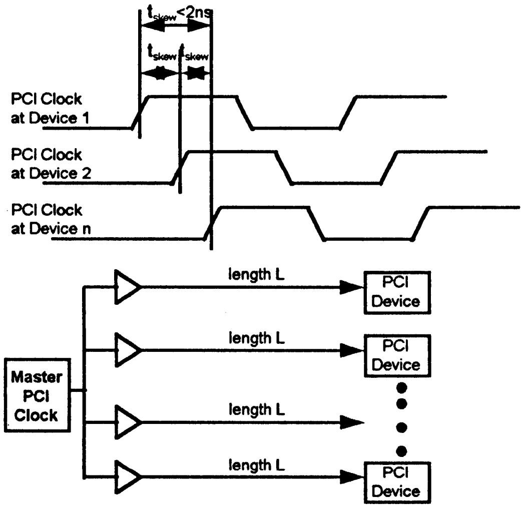
Split Power Plane Situations



- Routing over Split Power Planes Creates Impedance Discontinuities
- Route over same plane or cross over to route over Ground Plane
  (assuming physical dimensions are the same on other side)
- Or Capacitively tie together power planes with 0.01uF caps for every 4 signals

PCI Clock Distribution

- PCI Clocks are Point to Point
- Tskew < 2ns
- Severe Clock Skew at PCI devices is frequently cause of data corruption problems



PCI Expansion Card Layout Requirements

- Unloaded Characteristic Impedance = 60 to 100 ohms
- Propagation Delay = 0.15 to 0.19 ns/in
- Maximum Trace Length for PCI-32 bit signals < 1.5 in
- PCI CLK signal must be 2.5 in +/- 0.1 in

The PCI Signal Trace Calculator

Multiple Parameters Involved:

e, t, w, h, Z, t_{pd}, T_{PROP}, C, C_{load}, C_{via}, C_{stub}, $C_{i/o}$, Adjustment Factor

Several Equations:

$$Z = \frac{87}{\sqrt{e + 1.41}} \ln\left(\frac{5.98t}{0.8w + h}\right) \qquad t_{pd} = 0.08475 \sqrt{0.475e + 0.67 \text{ ns/in}}$$

$$\text{Adjustment factor} = \sqrt{\frac{(1.45 + C_{load})}{1.45}}$$

To Simplify Calculations, the **PCI Signal Trace Calculator**

Written in BASIC, 44 lines

PCI Signal Trace Calculator

Enter Dimensions in mils

```
'PCI_TRCE  PCI Circuit Board Trace Parameter Calculation Program
'INPUTS are TRACE WIDTH, TRACE THICKNESS, DIELECTRIC THICKNESS, DIELECTRIC
CONSTANT
'OUTPUTS are CHARACERISTIC OHMS, PROP DELAY, CAP/INCH, ADJST FACTOR,
'OUTPUTS are ADJUSTED OHMS, ADJUSTED PROP DELAY
'Jim Murashige, Adaptec  10/6/93
10  PRINT "TYPE THE TRACE WIDTH IN MILS, PRESS cr"
20  INPUT w
30  LPRINT "THE TRACE WIDTH IN MILS IS", w
40  PRINT "TYPE THE TRACE THICKNESS IN MILS, PRESS CR"
50  INPUT h
60  LPRINT "THE TRACE THICKNESS IN MILS IS", h
70  PRINT "TYPE THE DIELECTRIC SPACING IN MILS, PRESS cr"
80  INPUT t
90  LPRINT "THE DIELECTRIC SPACING IN MILS IS", t
100 PRINT "TYPE THE DIELECTRIC CONSTANT, PRESS cr"
110 INPUT e
120 LPRINT "THE DIELECTRIC CONSTANT IS", e
130 T1 = (5.98 * t) / ((.8 * w) + h)
140 T2 = LOG(T1)
150 T3 = SQR(e + 1.41)
160 Z = (87 / T3) * T2
180 PRINT "THE CHARACTERISTIC IMPEDANCE IS:", Z, "OHMS"
190 LPRINT "THE CHARACTERISTIC IMPEDANCE IS:", Z, "OHMS"
200 PROP = .08475 * SQR((.475 * E) + .67)
210 PRINT "THE PROPAGATION DELAY IS:", PROP, "NS/INCH"
220 LPRINT "THE PROPAGATION DELAY IS:", PROP, "NS/INCH"
230 CAP = (PROP / Z) * 1000
240 PRINT "THE TRACE CAPACITANCE/INCH IS:", CAP, "PF/INCH"
250 LPRINT "THE TRACB CAPACITANCE/INCH IS:", CAP, "PF/INCH"
260 LOADC = 11 + (CAP * 1.5)
270 ADJ = SQR(1 + (LOADC / CAP))
280 PRINT "THE ADJUSTMENT FACTOR IS:", ADJ
290 LPRINT "THE ADJUSTMENT FACTOR IS:", ADJ
300 ADJOHM = Z / ADJ
310 ADJPROP = PROP * ADJ
320 PRINT "THE ADJUSTED CHARACTERISTIC IMPEDANCE IS:", ADJOHM, "OHMS"
330 LPRINT "THE ADJUSTED CHARACTERISTIC IMPEDANCE IS:", ADJOHM, "OHMS"
340 PRINT "THE ADJUSTED PROPAGATION DELAY IS:", ADJPROP, "NS/INCH"
350 LPRINT "THE ADJUSTED PROPAGATION DELAY IS:", ADJPROP, "NS/INCH"
360 MAXTR = 10 / ADJPROP / 2
370 PRINT "THE MAXIMUM SPEEDWAY TRACE LENGTH IS:", MAXTR, "INCHES"
380 LPRINT "THE MAXIMUM SPEEDWAY TRACE LENGTH IS:", MAXTR, "INCHES"
390 LPRINT
400 LPRINT
410 PRINT
420 PRINT
430 GOTO 10
440 STOP
```

In Summary

- The PCI Signaling Environment Requires a Fresh Look at Circuit Interconnects
- Higher Speeds are Turning Circuit Boards into Design Components
- Physics and T_{PROP} Requirements Limit Bus Lengths
- Impedance Discontinuities Should Be Avoided, i.e.. Slot Riser Card Designs

Reference Sources

PCI Local bus Specification, Revision 2.0,

April 30, 1993; PCI Special Interest Group
(800)433-5177
(503)797-4207

"Treat circuit boards as design components in PCI-based systems"
EDN Magazine; November 23, 1994; page 111-116

PCI Electrical Design & Bus Layout

Jim Murashige
Applications Engineer
Adaptec
Mail Stop 195
691 S. Milpitas Blvd.
Milpitas, CA 95035
(408) 957-4813/7295 (fax)

Jim Murashige has been an applications engineer at Adaptec for the past 5 years. He Currently is working on PCI technology in the Adaptec ECX (Enterprise Computing) division. He is involved with Technical Marketing, Industry Relationships with BIOS & Chipset vendors, PCI compatibility, and performance Analysis. Received an BSEE from John Hopkins University, Baltimore, Maryland.

Reprinted Courtesy of EDN, November 23, 1994

- An examination of the PCI Electrical switching environment and how it is different from previous bus interfaces.

- The circuit board as a design element in PCI.

- Circuit layout suggestions and requirements from the PCI v2.0 specifications.

Treat circuit boards as design components in PCI-based systems

JIM MURASHIGE, ADAPTEC

By treating pc-board traces and device loads as components in a transmission line, you can create high-speed PCI-system designs that satisfy the PCI Specification Rev 2.0.

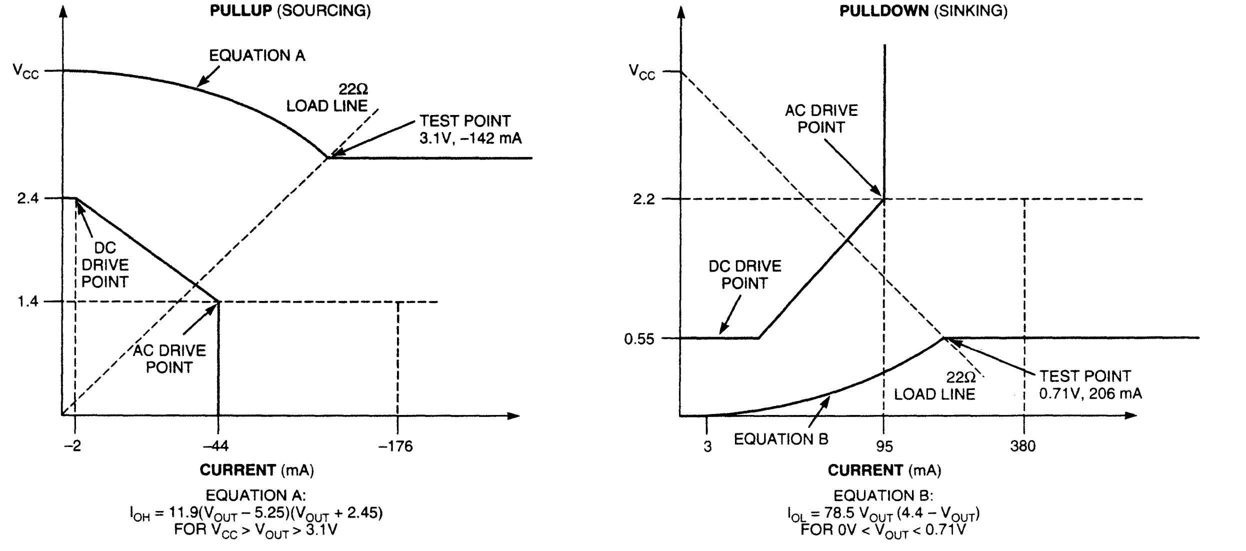
To obtain the highest levels of performance possible with a local-bus interface while using currently available technology, you should treat the layouts and designs for PCI motherboards and add-in cards as signal-transmission environments, as opposed to simple component-wiring interconnects. To help you in this endeavor, the PCI Specification Revision 2.0 introduces several concepts and highlights circuit effects that were formerly small enough to ignore.

The circuit board thus becomes an electrical element to consider in design. To design PCI circuits successfully, you should thoroughly understand and implement the principles and requirements in the PCI standard. Moreover, you should be well-versed in transmission theory. The PCI standard contains several major points about signal transmission.

The PCI Revision 2.0 spec defines signaling in 3.3 and 5V systems. To avoid confusion, we cover only the 5V environment here, though the principles involved apply identically to 3.3V systems. By definition, PCI components are CMOS devices with small input-leakage currents, although in the 5V signaling environment, they operate with TTL signal levels. Output-drive capability is not much of an issue in the steady-state, dc condition, as the outputs must supply only small input-leakage currents. Of great concern, however, is the output switching capability during logic transitions.

BBS The bibliography and software listings mentioned in this article are available on the EDN Readers' BBS. Phone (617) 558-4241 with modem settings 1200/2400 8,N,1 (9600 baud=(617) 558-4580). From the Main-System Menu enter *ss/freeware*. Then from the /freeware SIG menu enter *rkms855*.

FIGURE 1



To ensure sufficient—but not excessive—signal drive during ac switching, the PCI Rev 2 spec imposes minimum/maximum limits on current drive, for both the pullup (sourcing) and pulldown (sinking) transitions.

Fig 1 shows the minimum/maximum output-drive requirements for 5V signaling in the PCI Revision 2.0 spec. The curves show the minimum and maximum drive levels at given signal voltages, with the shaded areas acceptable. During logic transitions, the pc-board traces that interconnect PCI components appear as transmission lines, with ohmic characteristic impedances.

To drive a transmission line, the outputs must source or sink a minimum amount of current to ensure a large enough voltage step on the line, given the characteristic impedance. At the same time, however, it's necessary to limit the drive current to keep the reflected voltage wave within acceptable limits. The minimum and maximum ac-drive curves reflect these two considerations.

The 5V V-I drive curves in **Fig 1** show that the minimum impedance an output must be able to drive is 31.8Ω (1.4V/44 mA). For 3.3V systems, this figure is 37.5Ω. All PCI components must satisfy the minimum/maximum drive curves to ensure sufficient signal drive during ac switching. **Fig 2** shows the PCI signal characteristics of an Adaptec AIC-7870 PCI-SCSI controller.

PCI systems use "reflected-wave" switching, in which the initial voltage wave that travels down the transmission line isn't large enough to cause a logic transition. It must wait to be reinforced by the wave reflecting off the end of the transmission line. For this reason, the ends of the line are left unterminated. In reflected-wave switching, the device farthest away from the driving device switches first, followed by the nearest device, which switches last.

PCI defines a worst-case signal-propagation delay (t_{PROP}) of 10 nsec, which, in the light of the reflected-wave switching, mandates a maximum round-trip signal-propagation time down the PCI bus and back again of t_{PROP}=10 nsec. However, you can relax this requirement in systems operating with a PCI clock frequency lower than 33 MHz. You can also somewhat relax t_{PROP} if you can economize on t_{SKEW}. You can combine the t_{SKEW} budget with t_{PROP} for a total maximum round-trip delay of 12 nsec.

The pc board as a PCI-design component

The objective in PCI-system-board design is to minimize impedance discontinuities that cause signal degradation, while satisfying the characteristic-impedance and propagation-delay requirements. Characteristic impedance and propagation delay are functions of the intrinsic inductance and capacitance of the circuit-board traces. These parameters are, in turn, directly related to the physical geometry of the circuit board. Propagation delay and characteristic impedance are interrelated; changing one affects the other.

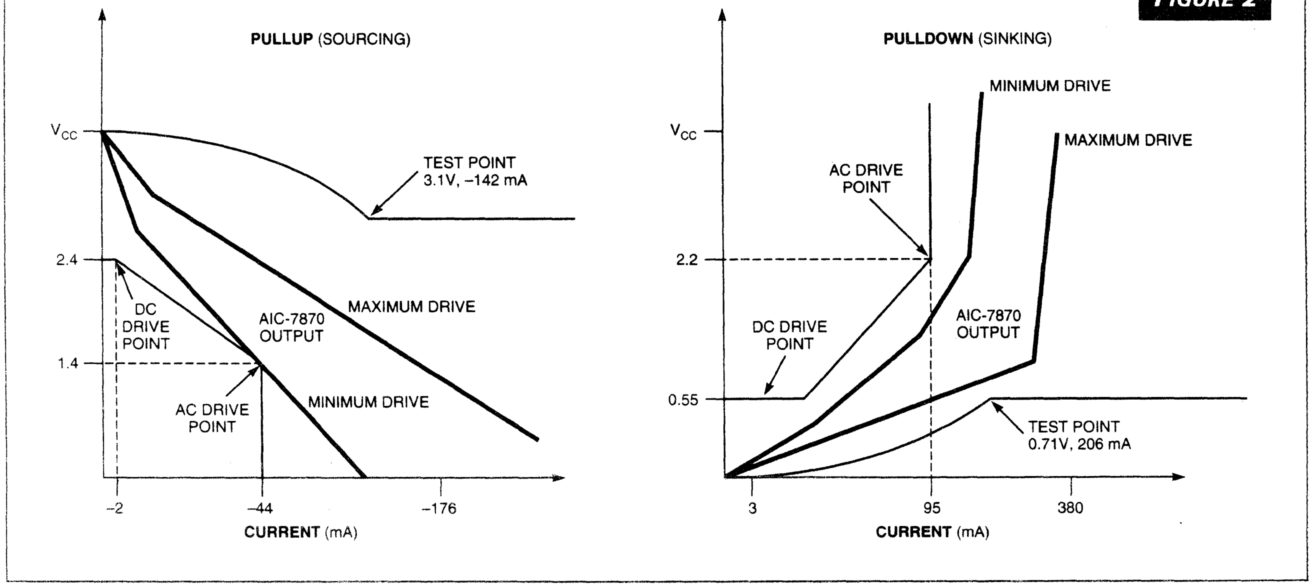
Trace inductance is a function of the cross-sectional dimensions of the circuit-board traces. Trace capacitance is a function of the trace surface area, physical proximity of the trace to a power or ground plane, and the type of laminate used. The cross-sectional view in **Fig 3** shows the physical dimensions that determine trace inductance and capacitance in the case of a signal over a power or ground plane (micro stripline).

It's difficult to calculate exact trace impedance because of the geometries involved. However, a workable formula for calculating the impedance of micro-stripline is:

$$z = \frac{87}{\sqrt{e + 1.41}} \ln\left(\frac{5.98t}{0.8w + h}\right)$$

where e is the dielectric constant of the medium, t is the dielectric thickness, w is the trace width, and h is the trace thickness. **Fig 4** shows an example of trace-impedance con-

FIGURE 2



Designers of PCI-standard circuits must ensure that the minimum/maximum current-drive capabilities of the circuits fall within the shaded areas, as is the case for Adaptec's AIC-7870 PCI-SCSI controller.

trol, in which the unloaded board-signal impedance calculates to about 102Ω.

Propagation delay is directly related to signal velocity (1/√LC), which in free space is the speed of light: 3×10^8 m/sec, or 11.81 in./nsec. A workable formula for calculating the propagation delay of a micro-stripline signal is:

$$t_{PD}=0.08475\sqrt{0.475e+0.67}\text{ nsec/in.}$$

For a G10 glass-epoxy circuit board, the dielectric constant is approximately 5. This yields a propagation delay (t_{PD}) of 0.148 nsec/in.

Racing along the PCI Speedway

As an aid to laying out motherboards, PCI suggests the PCI "Speedway" topology (**Fig 5**). The recommended Speedway dimensions are:

$$W_{SPEEDWAY}=\text{width}=0.6\text{ in. typ}$$
$$L_{STUB}=\text{stub length (Speedway-to-load)}=1.5\text{ in. max}$$
$$L_{LINE}=\text{line length (stub-to-stub)}=2\text{ in.}$$

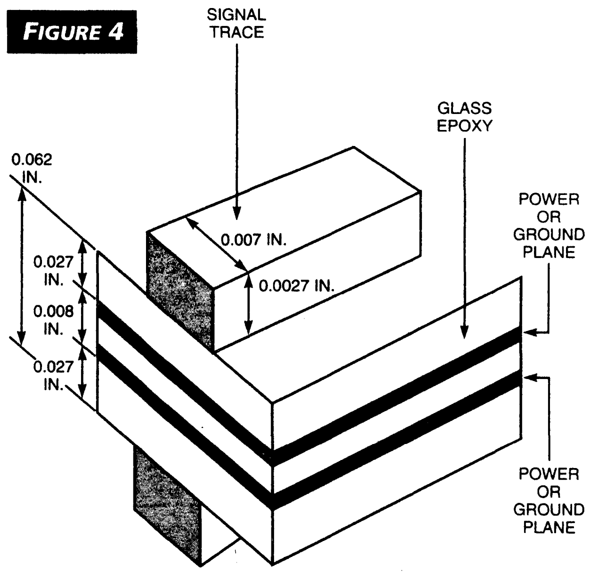
Devices are placed on both sides of the Speedway and staggered such that the physical device-stub-to-device-stub spacing is 1 in. If you follow the spacing recommendations, the Speedway appears electrically as a transmission line with evenly distributed capacitive loads (**Fig 6**). Device loads must be evenly distributed because, if they are physically grouped together, their loading effects appear as a lumped impedance discontinuity, resulting in signal-reflection problems.

The loading character of PCI devices is principally capacitive. If the devices are evenly distributed, then you can consider their capacitive loading effects to contribute to the transmission-line characteristics. Qualitatively, this additional transmission-line capacitance lowers the characteristic impedance and increases the propagation delay.

As an example of recalculating the characteristic impedance and propagation delay, return to **Fig 4**'s 102Ω circuit-board example. The signal-trace capacitance/in. depends on the physical trace width and the thickness of the circuit-board dielectric. **Fig 7** shows the relevant parameters.

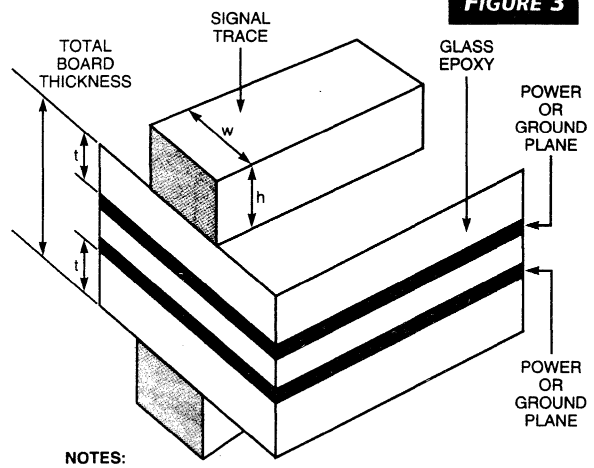
It's difficult to calculate capacitance/unit length directly

FIGURE 4



The formulas in the text, using the dielectric thickness, the dielectric constant of the board material, and the trace thickness and width, yield 102Ω characteristic impedance and 0.148-nsec/in. propagation delay for this board.

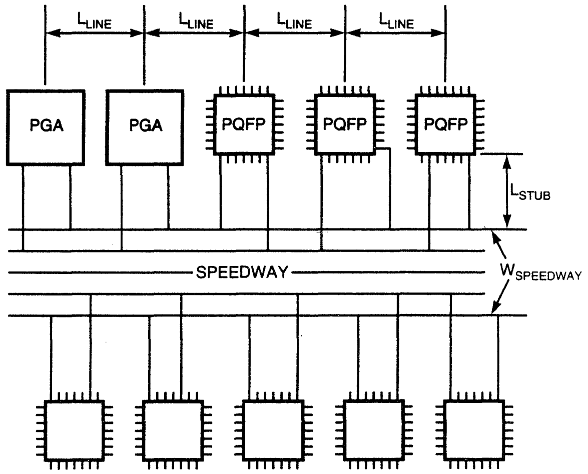
FIGURE 3



NOTES:
C – TRACE CAPACITANCE/IN. IS PROPORTIONAL TO w/t
L – TRACE INDUCTANCE/IN. IS PROPORTIONAL TO 1/(w + h)
CHARACTERISTIC IMPEDANCE/IN. IS $\sqrt{L/C}$
PROPOGATION DELAY/IN. IS $\sqrt{LC}$

Trace capacitance and inductance are functions of trace thickness and width, proximity to the power or ground plane, and the dielectric properties of the board material.

FIGURE 5



The PCI Speedway topology staggers devices along the bus at 1-in. intervals and avoids the lumped-impedance discontinuity that grouping the devices together would cause.

because of fringing effects. However, the relationship $C=t_{PD}/Z$ yields a usable figure. In the example of **Fig 7**, it works out to 1.45 pF/in. of trace length. Assuming you use a PCI Speedway layout with the dimensions in **Fig 8**, the additional capacitive loading per PCI device is:

$$C_{LOAD}=C_{VIA}+C_{STUB}+C_{I/O},$$

where C_{VIA} is the capacitance per feedthrough hole, C_{STUB} is the capacitance of the trace connecting the device to the Speedway, and $C_{I/O}$ is the capacitive loading per device.

Assuming that C_{VIA}=0.5 pF and a device load is 10 pF, C_{LOAD}=0.5 pF+(1.45 pF/in.×1.5 in.)+0.5 pF+10 pF=13.175 pF. Also assuming that the PCI devices are regularly spaced, you can consider the load to be distributed and additive to the trace capacitance of 1.45 pF/in. This assumption allows you to calculate an adjustment factor to apply to the originally calculated values for impedance and propagation delay.

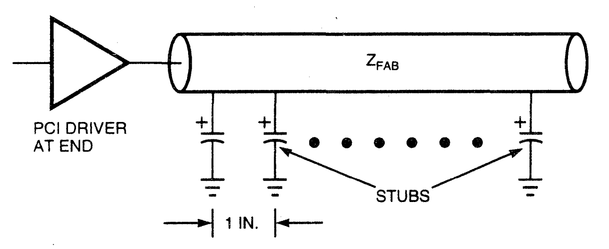
In this case, the adjustment factor is $\sqrt{1.45+C_{LOAD}/1.45}$=3.176. You revise the characteristic impedance downward and the propagation delay upward by this factor. You now obtain a loaded characteristic-impedance value of 32.1Ω and an adjusted propagation delay of 0.47 nsec/in. You can use these adjusted values to determine whether this PCI layout meets the PCI spec requirements.

As **Fig 1** shows, the trace impedance in a 5V signaling environment has a minimum spec of 31.8Ω, which our example satisfies. The round-trip signal-propagation time must be less than 10 nsec. Applying the revised propagation-delay figure yields a round-trip trace length of 21.29 in. The maximum allowable trace length is thus 21.29÷2=10.65 in.

To aid in signal-trace parameter calculation, the Basic program in the **Listing 1** calculates and prints all the derived parameter values and adjustment factors, using the formulae given in the text. Inputs to the program are the circuit-board trace dimensions, the dielectric thickness, and the dielectric constant of the board material.

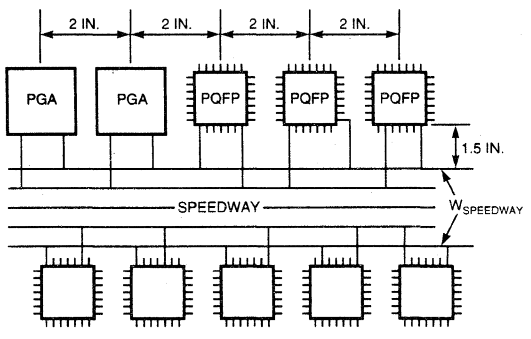
Trace and load capacitance has the general effect of reduc-

FIGURE 6



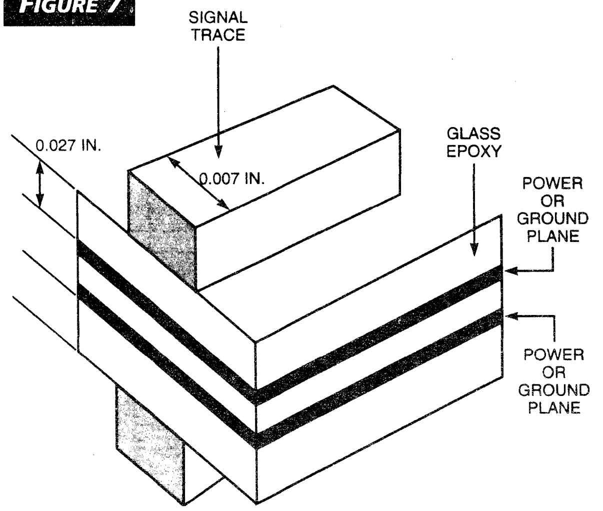
If you follow the spacing recommendations in the PCI spec, the Speedway resembles a transmission line with evenly distributed capacitive loads at each device stub.

FIGURE 8



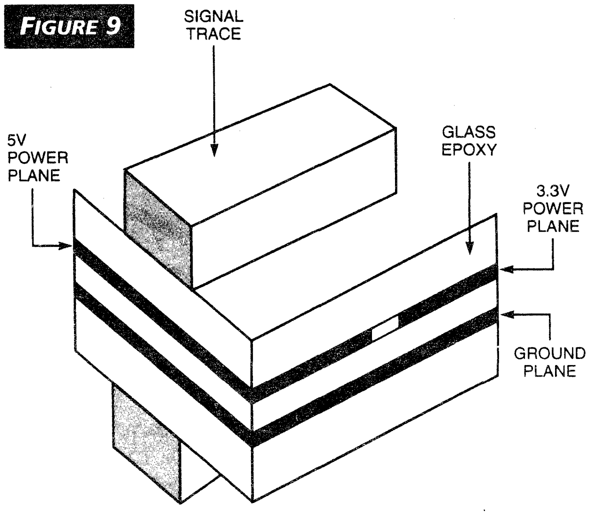
In this typical Speedway layout, the trace and device capacitances conspire to worsen the characteristic impedance and propagation delay of the design by a factor of more than 3.

FIGURE 7



Capacitance of the signal trace/in. depends on the trace width and the thickness of the circuit-board dielectric.

FIGURE 9



In mixed 3.3 and 5V systems, the power plane may be split, causing impedance discontinuities for signals running over the split.

ing signal velocity and lowering characteristic impedance. Naturally, minimizing trace and load capacitance is beneficial; it ensures signal integrity through higher signal velocity and higher impedance. Trace capacitance is directly proportional to the trace width, so narrower traces reduce capacitance and help signal transmission. However, it's difficult to reliably fabricate traces narrower than 5 mils while maintaining consistent characteristic impedance.

These analyses assume that signal traces run over a continuous ground or power plane. However, in dual 3.3 and 5V systems, situations arise in which the power plane is split, causing impedance discontinuities for signals running over the split. **Fig 9** shows a split-power-plane situation, in which you should route the high-speed signals to avoid the split or pass them through to the opposite side of the board and reference them to the ground plane. Failing these solutions, you should capacitively couple the two power planes, as the PCI spec recommends, using a 0.01-μF capacitor for every four signals crossing the split. Place the capacitors no farther than 0.25 in. from the signal crossing.

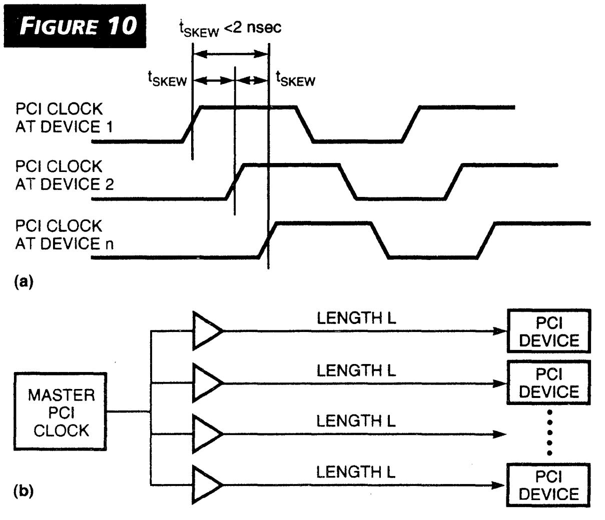
Don't daisy-chain clock lines

So far, we've dealt with the situation of PCI devices daisy-chained on the same bus, with signals propagating down the bus from device to device. The PCI clock source, however, needs to arrive at each device with a maximum t_{SKEW} delay of 2 nsec (**Fig 10a**). Because of this, you should not daisy-chain the PCI clock from device to device but rather route the main clock source directly to each device with equal trace lengths to minimize skew (**Fig 10b**).

The PCI Rev 2.0 spec does not mandate a trace-layout geometry for motherboard designs. The geometry is at the discretion of motherboard designers, provided the design meets the requirements for round-trip signal-propagation delay and loaded-trace impedance. The requirements for expansion-card designs are more stringent to ensure hospitable electrical characteristics. The unloaded characteristic impedance of signal traces on expansion cards is specified at 60 to 100Ω, with a propagation delay from 0.15 to 0.19 nsec/in. The maximum trace length for all 32-bit PCI signals is 1.5 in.; for 64-bit signals, it's 2 in. The PCI clock (CLK) signal must run 2.5±0.1 in. to minimize clock skew.

To summarize, treat the circuit board as a design component to achieve robust, reliable designs in high-speed, local-bus designs such as PCI. In particular, consider the board's characteristic impedance and propagation delay. To allow wide latitude in motherboard designs, the PCI Rev 2.0 spec imposes few requirements other than current drive and propagation delay. PCI expansion cards, on the other hand, have more stringent design requirements to ensure electrically compatible environments.

FIGURE 10



Clock skew in PCI systems must not exceed 2 nsec (a); therefore, it's wise to route clock signals directly from the clock source to each device (b), rather than in daisy-chain fashion.

LISTING 1—SIGNAL-TRACE PARAMETER CALCULATION

```
'PCI_TRCE  PCI Circuit Board Trace Parameter Calculation Program
'INPUTS are TRACE WIDTH, TRACE THICKNESS, DIELECTRIC THICKNESS, DIELECTRIC
'OUTPUTS are CHARACERISTIC OHMS, PROP DELAY, CAP/INCH, ADJST FACTOR,
'OUTPUTS are ADJUSTED OHMS, ADJUSTED PROP DELAY
'Jim Murashige, Adaptec  10/6/93
10  PRINT "TYPE THE TRACE WIDTH IN MILS, PRESS cr"
20  INPUT w
30  LPRINT "THE TRACE WIDTH IN MILS IS", w
40  PRINT "TYPE THE TRACE THICKNESS IN MILS, PRESS CR"
50  INPUT h
60  LPRINT "THE TRACE THICKNESS IN MILS IS", h
70  PRINT "TYPE THE DIELECTRIC SPACING IN MILS, PRESS cr"
80  INPUT t
90  LPRINT "THE DIELECTRIC SPACING IN MILS IS", t
100 PRINT "TYPE THE DIELECTRIC CONSTANT, PRESS cr"
110 INPUT e
120 LPRINT "THE DIELECTRIC CONSTANT IS", e
130 T1 = (5.98 * t) / ((.8 * w) + h)
140 T2 = LOG(T1)
150 T3 = SQR(e + 1.41)
160 Z = (87 / T3) * T2
180 PRINT "THE CHARACTERISTIC IMPEDANCE IS:", Z, "OHMS"
190 LPRINT "THE CHARACTERISTIC IMPEDANCE IS:", Z, "OHMS"
200 PROP = .08475 * SQR((.475 * E) + .67)
210 PRINT "THE PROPAGATION DELAY IS:", PROP, "NS/INCH"
220 LPRINT "THE PROPAGATION DELAY IS:", PROP, "NS/INCH"
230 CAP = (PROP / Z) * 1000
240 PRINT "THE TRACE CAPACITANCE/INCH IS:", CAP, "PF/INCH"
250 LPRINT "THE TRACE CAPACITANCE/INCH IS:", CAP, "PF/INCH"
260 LOADC = 11 + (CAP * 1.5)
270 ADJ = SQR(1 + (LOADC / CAP))
280 PRINT "THE ADJUSTMENT FACTOR IS:", ADJ
290 LPRINT "THE ADJUSTMENT FACTOR IS:", ADJ
300 ADJOHM = Z / ADJ
310 ADJPROP = PROP * ADJ
320 PRINT "THE ADJUSTED CHARACTERISTIC IMPEDANCE IS:", ADJOHM, "OHMS"
330 LPRINT "THE ADJUSTED CHARACTERISTIC IMPEDANCE IS:", ADJOHM, "OHMS"
340 PRINT "THE ADJUSTED PROPAGATION DELAY IS:", ADJPROP, "NS/INCH"
350 LPRINT "THE ADJUSTED PROPAGATION DELAY IS:", ADJPROP, "NS/INCH"
360 MAXTR = 10 / ADJPROP / 2
370 PRINT "THE MAXIMUM SPEEDWAY TRACE LENGTH IS:", MAXTR, "INCHES"
380 LPRINT "THE MAXIMUM SPEEDWAY TRACE LENGTH IS:", MAXTR, "INCHES"
390 LPRINT
400 LPRINT
410 PRINT
420 PRINT
430 GOTO 10
440 STOP
```

Reference

1. PCI Local Bus Specification, Revision 2.0, April 30, 1993; PCI Special Interest Group

Author's biography

Jim Murashige is an applications engineer at Adaptec (Milpitas, CA), where he has developed SCSI-standard chips for four years. He's in charge of technical marketing, performance analysis, and OEM design assistance. Murashige earned a BSEE at Johns Hopkins University, Baltimore. His spare-time pursuits include home improvement, travel, and entrepreneuring in the home-automation market.

Customization with PCI Functional System Block

Sam Sanyal
VLSI Technology Inc.,
1109 McKay Drive, MS# 33
San Jose, California 95131
sanyal_s@sanjose.vlsi.com

Brad Bailey
VLSI Technology Inc.,
8375 South River Parkway, MS# 275
Tempe, Arizona 85284
bailey_b@vlsiphx.vlsi.com

ABSTRACT

*In this paper we present the VLSI **VCF94100** Peripheral Component Interconnect(PCI) FSB™ (functional sytem block).*

The VCF94100 FSB core allows a peripheral component to be connected to a PCI Local bus with a minimal amount of design effort. The VCF94100 provides full compliance as a Master and/or Target as defined in the PCI Local Bus Specification, Revision 2.0

A key feature of the VCF94100 FSB™ element is the Configuration Logic Block. The design of the Configuration Logic Block is such that it may be easily and quickly modified to meet the individual configuration space requirements of each user.

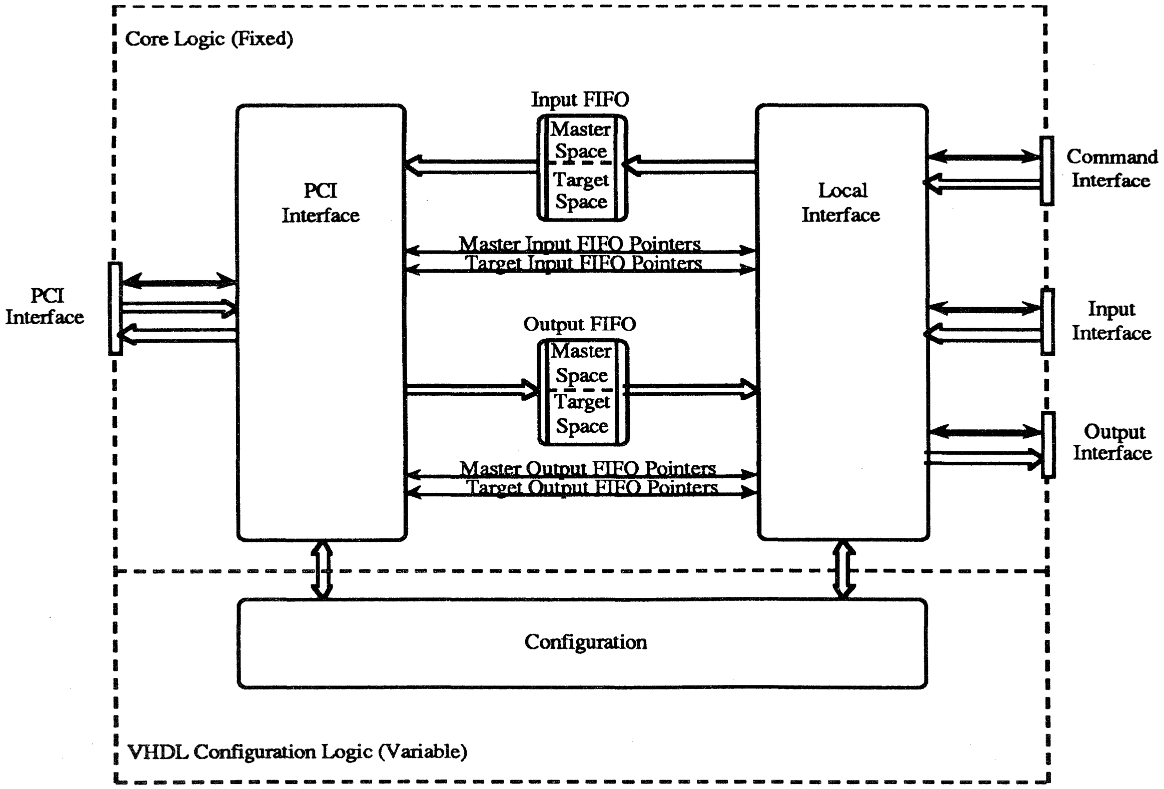
The VCF94100 FSB™ element is a high performance stand-alone block that is easily integrated into any ASIC or ASSP device.

INTRODUCTION

The VCF94100 provides a flexible, high performance solution for interfacing to the PCI bus. The VCF94100 incorporates the following features:

- Fully compliant with PCI Specification, Revision 2.0 for a 32 bit peripheral device
- Can act as a Bus Master, Bus Target, or both
- Supports full range of memory and I/O operations, from single transfers to infinitely long transfers in either burst mode or wait
- Provides an infinitely adjustable configuration space which is uniquely defined by each user
- PCI clock and on-chip clock isolation. The VCF94100 can support the required 0-33 MHz clock frequency on the PCI Interface, and will support up to 50 MHz on the Local Interface.

PCI FSB Block Diagram



FSB is a trademark of VLSI Technology, Inc.

- Area efficient design which utilizes roughly 12K gates with a typical Configuration Logic Block

INTERFACES

The VCF94100 has four major interfaces:

- PCI Interface
- Local Input Interface
- Local Output Interface
- Command Interface

The *PCI Interface* contains all the signals needed to connect the VCF94100 up to a PCI bus. The signals are broken into groups that interface to I/O pads. For example, the PCI IRDY signal is represented as an IRDYin and an IRDYout signal on the VCF94100.

The *Local Input Interface* supports the transfer of on-chip data to the resynchronization FIFOs. This interface is used during master write and target read transactions. The interface is purely synchronous with respect to the on-chip clock.

This is a simple interface that includes an address bus, a data bus, and some control signals. The bus functions similar to that of a DMA, where the VCF94100 is responsible for initiating each transaction. The interface can transfer data at a maximum rate of one DWORD per clock cycle. If required, wait states can be injected by either the VCF94100 when the FIFO is full, or by on-chip logic when data to be transferred is not yet available.

A signal is driven from the VCF94100 to indicate whether the current bus activity is associated with a master or a target transaction.

The *Local Output Interface* supports the transfer of data from the resynchronization FIFOs to on-chip resources. This interface is used during master read and target write transactions. With exception to the direction of data flow, this interface operates in the same manner as the Local Input Interface described above.

The *Command Interface* is used to transfer the information needed to setup the VCF94100 for a master transaction. This interface operates purely as a slave type interface, with the on-chip logic initiating the data transfer. The interface includes a data-in bus, two select inputs, and some control signals.

There are three words that get loaded into the VCF94100 to initiate a master transaction. These words are:

- PCI Address
- Local Address
- Control

The on-chip logic will drive the two select lines to indicate which of the three words are being presented to the data-in bus. The loading of either the PCI Address or the Local Address is optional. The loading of the Control word actually starts the processing of a master transaction.

The *PCI Address* word is a pointer into PCI address space. The *Local Address* word is a pointer into the on-chip memory space. If these do not get loaded with a control word, then the master transaction will proceed using the existing residual address values that were left over since the last master transaction. An example of how this could be beneficial would be the case where source data is held in non-contiguous blocks and destination data is to be stored in contiguous blocks.

The *Control* word contains the remaining miscellaneous control information for the master transaction. Included in this word are:

- Transfer Count
- PCI Command
- Function Space
- PCI/Local Address Increment
- Continuous

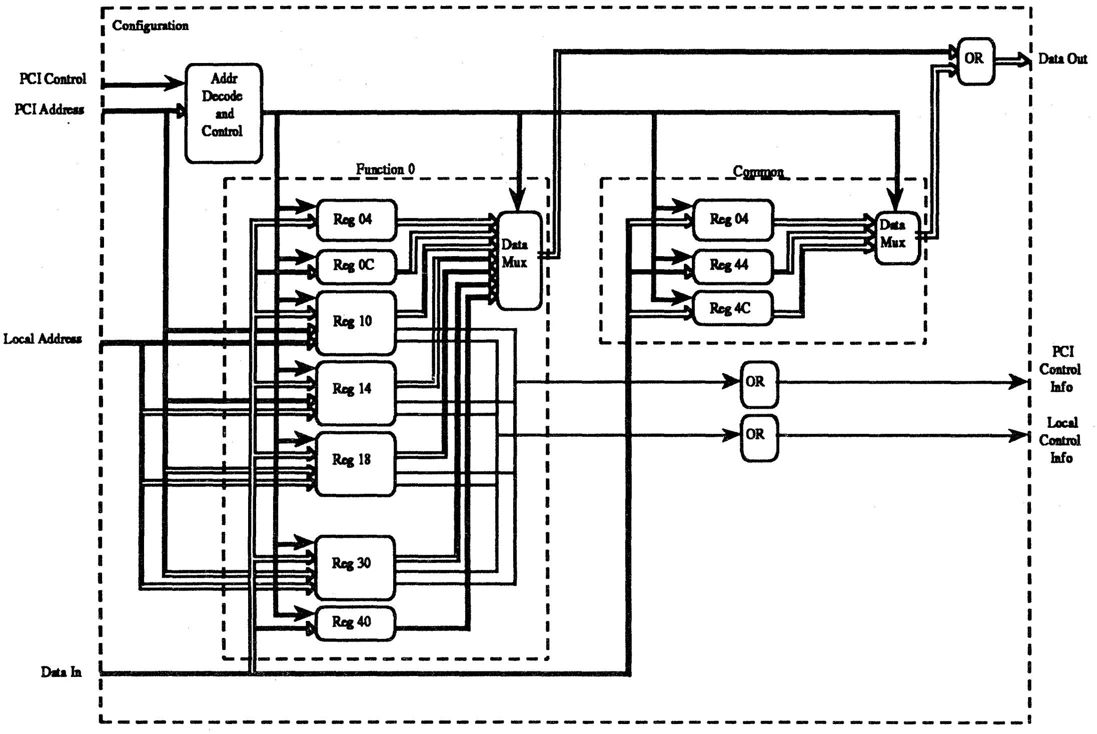
The Transfer Count is the byte count for the transaction. The PCI Command is per the PCI Specification Revision 2.0 for PCI command types.

The Function Space is a pointer into one of eight functions within configuration space that is to be used when performing the master transaction. This is only used for VCF94100s that are configured as multi-functional devices. There are items within each function that dictate how certain items happen during a master transaction. This field allows the VCF94100 to validate whether a requested master transaction meets the requirements held in the associated function space. An good example of this is the Bus Master bit, where each function has its own Bus Master bit.

The PCI/Local Address Increment was designed in the interest of supporting constant address FIFO devices. When set, the address presented to the associated interface will not be incremented. For the case of PCI accesses, this will only be apparent during Disconnect-Retry cycles since the bus implements implied addressing.

The Continuous bit provides an alternative method to controlling the length of a transaction. When set, the transfer count is ignored, and the transaction continuous until the on-chip logic asserts a signal indicating the completion of the transaction.

Configuration Logic Block Diagram



INTERNAL FUNCTIONS

The VCF94100 design is subdivided into two distinct blocks:

- Configuration Logic Block
- Core Logic Block

The Configuration Logic Block is considered the modifiable portion of the design, while the Core Logic Block is the fixed portion of the design. A user of the VCF94100 will custom tailor the Configuration Logic Block according to their particular system requirements.

Configuration Logic Block

Each device that connects to the PCI bus will have a unique set of memory, I/O or ROM resources. For example, some elements may have only one memory range, which would require the use of only one Base Register. Multiple memory ranges would require multiple Base Registers. In addition, the physical size of each resource being offered is uniquely encoded in the form of fixed lower bits within each Base Register.
There are other fields in the configuration space, such as the Vendor ID and Device ID fields, that are also unique to each design.

In the interest of creating only the logic necessary to implement the required functions of a unique configuration space, the VCF94100 configuration space is constructed entirely by the user in the form of VHDL source code. A sample of a typical configuration space is provided to the user. To create a unique configuration space, the user simply modifies this sample block as needed. The new code is then synthesized and joined to the already synthesized and laid-out Core Logic Block.

It is interesting to note that the PCI timing and protocol information is contained entirely within the Core Logic Block. Therefore the ability of the VCF94100 to remain fully PCI compliant is not jeopardized by any actions that may be taken by the user to create a new configuration space.

Configuration Support:

The VHDL-based Configuration Logic Block provided to the user has examples of how to
i
implement all the features of the PCI specified configuration space. This includes:

- Multiple Base Registers, including examples for memory, I/O and ROM
- Multiple Functions, from one to eight functions
- Status and Command registers
- All hardcoded bits featured as VHDL constants in a single package file
- Zero Wait State configuration access

Additional Support:

The Configuration Logic Block actually supports more than just those items identified for configuration space in the PCI specification. There are additional items that the user can adjust that further characterize how the VCF94100 operates. These items are:

Target I/O Transaction Byte Alignment:

The PCI specification is very specific about how a target responds to an I/O access with respect to the alignment of the lower two address bits and the byte enables. An illegal combination of the two, such as a byte enable pattern implying a lower address that specified in the two address bits (i.e. 0x00000001 and 0000) is considered an error and must be rejected.

The VCF94100 provides further capabilities. The user may specify in the configuration space how the VCF94100 is to respond to subsets within a byte enable and address pattern. For example, the user may declare an I/O space that will accept only a byte enable pattern of 1100 (assuming the proper lower address bits). A byte enable pattern of 1110 or 1101 may or may not be considered an error depending on how the user sets up the Configuration Logic Block. This logic is unique for each Base Register used to represent I/O space.

PCI to Local Address Translation

The VCF94100 allows the user to provide address translation between the PCI Interface and the Local Interface. A specified number of upper address bits from the PCI address get translated to create a new local address. This translation is unique for each Base Register.

Local Address Increment

By default, the VCF94100 will increment the local address by four for each DWORD that is transferred. The user may choose not to increment the local address if the device that is attached to the Local Interface operates in a manner similar to a constant address FIFO. This ability is unique for each Base Register.

Target Latency Timer

The Target Latency Timer provides an additional means of controlling the PCI bus bandwidth required by the VCF94100. This timer counts the number of PCI clock cycles that occur between data transfers. If the number of clock cycles between a data transfer exceeds the value programmed into this timer, the VCF94100 will initiate a Disconnect-Retry cycle.

Core Logic Block

The Core Logic block can be further subdivided into three more blocks. These are:

- PCI Interface
- Local Interface
- PCI/Local Resynchronization

PCI Interface:

The PCI Interface is responsible for supporting the flow of data between the PCI bus and the resynchronization FIFOs. All PCI timing and protocol information is contained within this block. This block is a fully synchronous design that is based on the PCI CLK signal. The sub-blocks of this design are:

- Master/Target State Machines
- Input/Output Data Paths
- Master/Target FIFO Control
- Parity Generation and Checking
- Master/Target Latency Timers

The *Master/Target State Machine* blocks are the heart of the PCI Interface. These blocks provide control to the Data Path and FIFO Control blocks. Inputs from the Latency Timer blocks are used to further determine responses. The Target State Machine has the added responsibility of controlling data transfers between the PCI bus and the Configuration Logic Block.

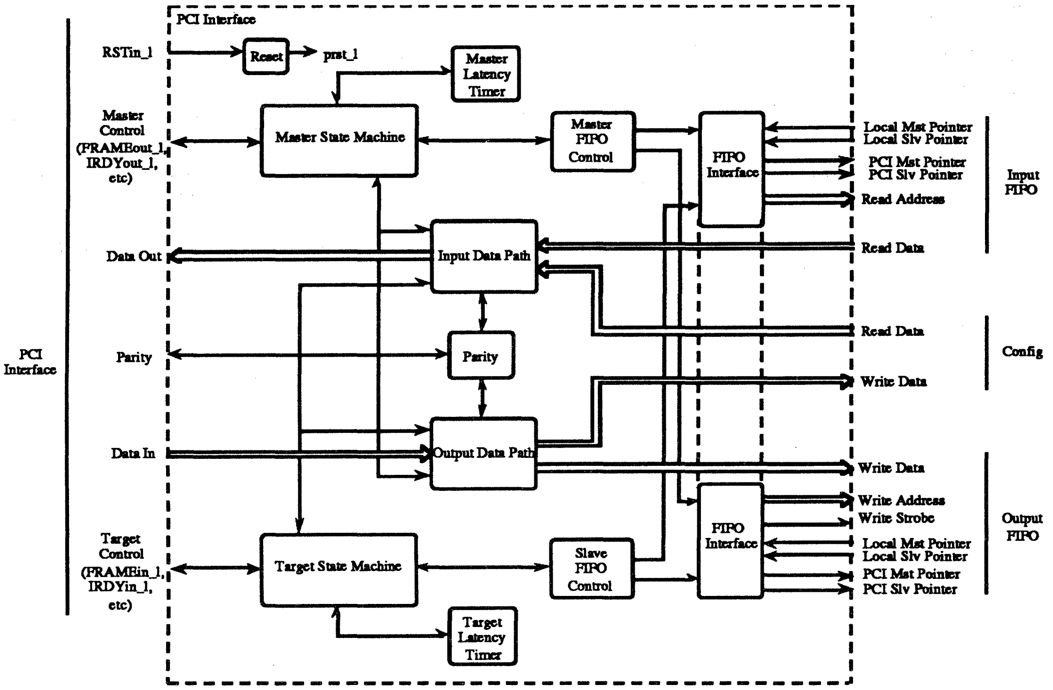
The *Input/Output Data Path* blocks provide a path for data to flow between the FIFOs and the PCI bus. In addition, configuration data is merged into or drawn from these blocks. Addressing information is contained within these blocks, with the Input Interface holding the target address and the Output interface holding the Master address.

The *Master/Target FIFO Control* blocks determine how data gets written to or read from the resynchronization FIFOs. A portion of this block is responsible for allocating cycle time between master and target transactions, while the other portion of this block keeps track of the status of each FIFO.

The *Parity Generation and Checking* block handles the generation and checking of parity on the PCI bus.

The *Master/Target Latency Timer* blocks control access times to the PCI bus. Count down information for each timer is obtained from the Configuration Logic Block. The master latency timer guarantees a minimum PCI access time as a master, and the target latency timer shuts down a target transaction after a pre-determined number of clocks have expired.

PCI Interface Block Diagram



Local Interface:

The Local Interface is responsible for supporting the flow of data between the on-chip resources and the resynchronization FIFOs. This block is a fully synchronous design that is based on the on-chip clock. The sub-blocks of this design are:

- Mode Control
- Master/Target Flow Control
- Master/Target Registers
- Input/Output Control
- Input/Output Data Path
- Input/Output FIFO Control

The *Mode Control* block is responsible for allocating the input and output resources within the VCF94100 to either master or target transactions. Recall that because of the clocking differences between the PCI Interface and the Local Interface, it is possible that the Local Interface could be tasked to simultaneously handle both a master and a target transaction. The Mode Control block allocates the resources needed for each type of transaction. For example, the Mode Control block will allow a simultaneous target read and a master read transaction to occur since one uses the input data path and the other uses the output data path. The Mode Control block not allow a simultaneous master read and target write transaction since both types require the use of the output data path.

The *Master/Target Flow Control* blocks handle the logistics associated with starting, maintaining, and stopping the transaction. The Master Flow Control block has the additional task of supporting the Command Interface.

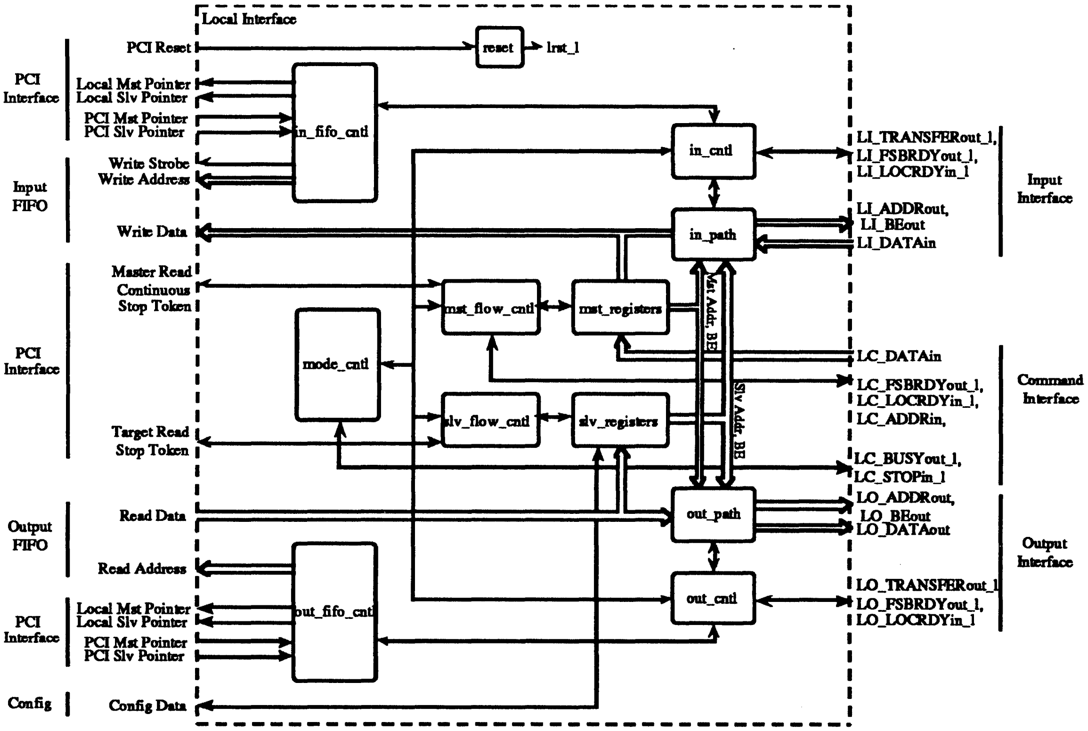
The *Master/Target Register* blocks hold the registered information associated with a transaction, such as the address counters, the byte enable registers, and the transfer count register.

The *Input/Output Control* blocks provide the lower level control needed to perform the actual data movement. These blocks feed information into the Data Path blocks and the FIFO Control blocks.

The *Input/Output Data Path* blocks provide the data path between the Local Interface and the FIFOs. These blocks contain the registers and miscellaneous logic needed to capture data from either the FIFOs or the Local Interface.

The *Input/Output FIFO Control* blocks determine how data gets written to or read from the resynchronization FIFOs. Information relative to the status of a FIFO (i.e. empty, almost empty, almost full, full) is generated here and sent to the Input/Output Control blocks.

Local Interface Block Diagram



PCI/Local Resynchronization:

The VCF94100 has the ability to provide complete clock isolation between the PCI and the Local interfaces. This feature off loads the user from dealing with resynchronizing data between the PCI bus and on-chip resources. This also proves beneficial in light of the specified requirement that the PCI clock can potentially be stopped or slowed down at any time (i.e. 0 - 33 MHz). The resynchronization logic isolates the on-chip circuitry from being affected by this condition.

The VCF94100 incorporates two dual-port RAMs and some control logic to implement a pair of FIFOs between the two interfaces. One FIFO supports the flow of data and control from the Local Interface to the PCI Interface, and the other FIFO supports the opposite flow.

Each FIFO is further subdivided into master space and target space. The two spaces are handled independently of each other by the corresponding master or target interface logic. This scheme allows master transactions to be performed completely independent of target transactions.

The FIFOs are used to pass controlling information between the two interfaces. For example, a special control word will be passed from the PCI Interface to the Local Interface at the start of each target transaction. This control word contains the translated address and the PCI command of the transaction. When a target transaction is completed, an indication of such is passed as control information within the FIFOs.

In a similar manner, the PCI Address and Control Command Interface words used to setup a master transaction gets passed from the Local Interface to the PCI Interface as command words in the input FIFO.

Additional circuitry is included that enables each interface to keep track of the status of the FIFOs. Several sets of hardware pointers are exchanged between the interfaces that indicate exactly where the opposing interface is in regards to processing FIFO information. For example, an interface waiting to write information into a FIFO will only do so when it has an indication from the opposing interface that the FIFO has room to be written to. In a similar manner, an interface will only read from a FIFO when there is an indication that there is new data within the FIFO to be read.

CONCLUSION

The VCF94100 is a highly integrated PCI interface solution that is flexible enough to be used in many different applications. The VCF94100 was created to make the task of interfacing to the PCI bus as simple for the designer as possible. Many of the issues that would concern a designer have already been resolved within the VCF94100. The use of a modifiable VHDL configuration space further enhances the users ability to quickly provide a complete single block solution.

ACKNOWLEDGMENTS

We would like to thank CPD marketing group specially Runbir Singh, Director, CPD Business Unit, Brad Kitson, MARCOM Group and Legal Department for helping us in completing the paper on time.

REFERENCES

[1] PCI reference manual

[2] VLSI VC94100 FSB^{TM} data sheet

AUTHORS BIOGRAPHY

Sam Sanyal

Sam Sanyal is a Staff Applications Engineer at VLSI Technology In., Sam earned his BSEE from Texas A& I University (Texas A&M University), MSEE from Cal Poly University and B.Sc from Calcutta University. Sam has worked with various Microprocessors for the past fourteen years.

Sam is involved in Indian Music (compose and sing) and community work. He likes Cricket, Soccer and Basket Ball game.

Brad Bailey

Brad Bailey is a Senior Design Engineer at VLSI Technology, Inc. Brad Bailey graduated from the University of Arizona in May 1985 with a degree in Electrical Engineering with an emphasis in Computer Science Engineering. He has been actively working as an ASIC/Board level designer for the past 10 years. His focus has been in the area of embedded control design.

Brad likes to race cars, drink beer, and go four-wheeling.

PCI BUS CORE DESIGN REDUCES PRODUCT COST, TIME TO MARKET

Frank J. Creede
President
Logic Innovations, Inc.
6205 Lusk Boulevard
San Diego, California 92121
(619) 455-7200

ABSTRACT

Product cost and time to market dictate a PCI bus product's potential for success. Product cost is driven by the quantity and cost of each component which comprises the product. It is safe to say that the first available and most cost effective product will have a preponderance for market success. The PCI Bus stringent AC/DC loading and drive requirements, as well as the six man month development effort required, works against both a PCI Bus product's cost and time to market. A turnkey, PCI Bus synthesizable core provides a quick turnaround design solution ready for immediate implementation in a cost effective ASIC or even a FPGA. A tested and silicon proven PCI Bus core implemented in VHDL/Verilog jumpstarts a PCI Bus product development with the confidence that only the application specific logic requires new engineering effort. The final ASIC implementation can yield an unmatched PCI Bus interface for $10.00 to $30.00 in volume.

THE PCI BUS INTERFACE PROBLEM

There are two aspects to architecting a PCI Bus interface. First, the very specific PCI Bus loading and drive requirements must be satisfied. Second, the PCI Bus is a high speed, flexible bus interface which requires significant engineering effort (six man months for either a Master or a Target). These aspects must be considered when developing a PCI Bus product development plan.

The challenge behind developing a PCI Bus product is not simply the engineering task at hand. The selection of the appropriate development approach will insure that a timely, cost effective PCI Product will be developed.

THE PCI BUS ALTERNATIVES

There are several PCI Bus implementation alternatives:

1 - Use an off the shelf, integrated peripheral device with a PCI Bus interface.

2 - Use a PCI Bus interface device in conjunction with a peripheral controller.

3 - Use a Mega-PAL and discrete logic in conjunction with a peripheral controller.

4 - Use a custom ASIC or FPGA implementation which integrates the PCI bus interface and the peripheral function into a single device.

Each alternative provides benefits and drawbacks; however, we are concerned with those issues which will inevitably determine the time to market and the final product cost.

Alternative # 1, an off the shelf, integrated peripheral device with a PCI Bus interface, is the no brainer. Should a designer chose a device which has the PCI Bus interface built in, the PCI Bus design becomes a connect the dots exercise. The time to market is the best with this approach. However, the product's cost and features will be similar to other PCI products which use the same device.

Alternative # 2, a PCI Bus interface device in conjunction with a peripheral controller, provides short time to market at the expense of product cost. PCI interface chips are not only costly, but peripheral interface logic will be required to connect to the peripheral controller or custom interface. The result is a multichip, more costly PCI Bus product.

Alternative # 3, a Mega-PAL and discrete logic in conjunction with a peripheral controller, provides both a longer product development cycle and a multichip, more costly PCI Bus product implementation. The Mega-PAL approach will require several man months of design effort to implement a minimal PCI Bus interface. Due to the small size of the Mega-PAL, significant custom peripheral interface logic will not fit within the device (in addition to the PCI Bus interface).

Alternative # 4, a custom ASIC or FPGA implementation, is the best approach for quick turn around and for minimized product cost. Based upon the availability of a silicon proven Verilog/VHDL synthesizable core, the designer need only add the peripheral interface logic to the ASIC design to complete the lowest chip count, most cost effective approach.

THE ASIC/FPGA ALTERNATIVE

With the availability of a proven PCI Bus core design, the development risk falls on the peripheral interface logic. Since many existing applications are moving to the PCI Bus from other environments, the peripheral interface logic for many users is a slam dunk.

Why use a PCI core design? Three reasons. First, the PCI Bus implementation itself will require three or more man months of development effort. This development involves both effort and risk. Second, the more than twenty-seven scenarios specified by the PCI Special Interest Group requires an equal if not greater effort in the validation of the design. Similarly, this task involves both effort and risk. Third, the use of a PCI synthesizable core will undoubtedly yield the most cost effective of all alternatives mentioned. The use of a custom ASIC will incur only the silicon cost for the features needed. The resulting design will yield a single chip PCI interface with peripheral controller, RAM, etc. All PCI Bus interface, user required logic and peripheral interface functions will reside within a single device.

FEATURES OF A PCI BUS SYNTHESIZABLE CORE

A good PCI bus synthesizable core should be a silicon proven, rigorously verified design. The core must be verified against the all scenarios of the PCI SIG compliance checklist. Finally, a royalty free unlimited use core allows for the design to be reused without fees tied to the number of designs the core is used in. In addition, the designers of the PCI bus core should have a demonstrable record of developing efficient, optimized designs which synthesize into high speed, minimal complexity implementations. Synthesizable core designs created by "model farms" are often inefficient, and contain enormous state machines which require modification to run at speed after they have been synthesized.

The following is a list of features for both a Master and a Target PCI core designs.

PCI Master Features

- Verilog or VHDL Design Compliant with IEEE 1076 and IEEE 1164 Extensions.
- PCI Bus Master Interface.
- I/O and Memory Space Supported.
- Full Speed Burst Memory Transfer Support (132M Bytes/Sec).
- Address and Data Parity Generation and Checking.
- PCI Interrupt Support.
- Custom Local Interfaces Provided:

  - FIFO
  - Dual Port SRAM
  - Single Port SRAM with Dual Port Arbiter
  - User Defined

- Validated to All Master Scenarios of PCI SIG Compliance Checklist.
- Approximate 7,000 Gate Implementation.

PCI Target Features

- Verilog or VHDL Design Compliant with IEEE 1076 and IEEE 1164 Extensions.
- PCI Bus Target Interface.
- I/O and Memory Space Supported.
- Full Speed Burst Memory Transfer Support (132M Bytes/Sec).
- Address and Data Parity Generation and Checking.
- PCI Interrupt Support.
- PCI Configuration Space Registers:

  - Device I. D., Vendor I. D.
  - Status, Commands
  - Class Code, Revision I. D.
  - Memory Base Address
  - I/O Base Address
  - Interrupt Line
  - Interrupt Pin
  - User Defined.

- Custom Local Interfaces Provided:

  - FIFO
  - Dual Port SRAM
  - Single Port SRAM with Dual Port Arbiter
  - User Defined

- Validated to All Target Scenarios of PCI SIG Compliance Checklist.
- Approximate 5,000 Gate Implementation.

CONCLUSION

The use of a PCI Bus synthesizable core in an ASIC or large scale FPGA for PCI Bus applications will provide the fastest development cycle with the most cost effective final product. For applications which require more than 20,000 gates, an ASIC or custom silicon device built around the synthesizable PCI core will provide the best solution. For implementations which require 20,000 gates or less, a large scale PCI compliant FPGA can be used for prototyping in short timeframe (a matter of 2-3 months). The FPGA may be converted into an ASIC for a cost reduced final product.

Using FPGAs for Peripheral Component Interconnect (PCI) Interface Designs

Authors: Brian Small, Kevin Yee, Charles Geber
QuickLogic Corporation
2933 Bunker Hill Lane
Santa Clara, CA 95054

Abstract

Peripheral Component Interconnect (PCI) is a recently defined standard Local Bus for high-speed processors and peripheral controllers. PCI is intended to meet the local bus requirements of next generation high-performance computer systems for several years, and has been adopted by several processor architectures and numerous system integrators. Digital Equipment (DEC Alpha), Motorola (PowerPC), and Intel (Pentium) are only a few of the industry players who have embraced this new bus interface for their processors and future desktop computers. This paper suggests methods of choosing an FPGA for PCI designs, and describes a complete PCI interface implemented in a single QuickLogic QL24x32B FPGA. The large logic and pinout capabilities of the QL24x32B device are key to providing the necessary interface functionality in a single FPGA device. In additions, the extremely fast I/O pad and internal logic can accommodate the stringent system timing requirements of the 33 MHz PCI bus. The design presented uses only half of the QL24X32B's resources, leaving room for additional system functions, like DMA logic. This paper will demonstrate how QuickLogic's high speed, pinout flexible, PCI compliant FPGA's can be used in PCI Applications, especially when the designer wants to integrate additional logic functions to a standard PCI Interface.

Choosing the right FPGA

Device Characteristics

PCI interface designs have difficult timing and board layout requirements. Picking an FPGA that can meet these requirements can be challenging. However, there are a few minimal criteria which, if used, can make the design much less painless - and faster to market (after, all - why use an FPGA if design-time isn't your primary focus?).

First of all, determine if the PCI design really requires an FPGA. There are many PCI designs whose requirements may be met by existing PCI chip-sets. However, a good portion of PCI designs have very stringent board-space requirements, so it would be more effective to use a higher density FPGA which would contain the PCI interface logic, and many other pieces of the entire board design.

Choose a high-density FPGA which will still meet the PCI timing restrictions. One such example is the QL24X32B (8000 gate) FPGA, which can easily meet and exceed the 33 MHz clock requirement, and 11 ns clock-to-output requirements of PCI designs.

Make sure the FPGA can interface with the PCI-bus directly. In other words, the FPGA cannot have I/O pin capacitance of greater then 10pf for pins attached to the bus, and the I/O drive must be able to meet the stringent PCI bus AC timing characteristics. These characteristics are outlined in the PCI Local Bus Specification. The difficult aspect of these AC timing characteristics is that they involve the drive capability of an I/O pin as the pin is changing value, not the static DC drive of a pin at a specific voltage. Check carefully that the FPGA you use meets the required bus driving requirements. The QL24X32B device from QuickLogic is one such device which meets these requirements.

One very important and often overlooked criteria for FPGA's in PCI designs is custom pinout support. Custom pinouts are needed in PCI designs to keep board traces short to reduce noise on the sensitive PCI bus. While many FPGA's support custom pinouts, the fixed pinout makes design changes difficult because routing resources on the devices are limited. The ideal FPGA device to use for PCI designs has guaranteed routability, even when a custom pinout is used. Quicklogic guarantees that a design with a custom pinout will always be routable, even across design iterations. QuickLogic's QL24X32B device has 172 available user I/O, making it easy to choose a pinout that will meet the difficult board-trace limitations of PCI bus signals.

FPGA Design Tools

FPGA designs are often riddled with difficulty in using design tools. In many FPGA design tools, auto-placement and routing of a device may not complete, or timing will change from one place and route to another - making the design process frustrating. All these factors make using traditional FPGA's for PCI design a tedious process, seemingly opposed to the goal of fast time to market.

The design of a PCI Interface requires a design entry environment focused on high-speed design results, with flexibility of design.

Ideally, the design tool allows easy mixing of different modes of design entry. HDL (Hardware Description Language) and schematic entry both need to be supported, because some areas of a PCI design (state machines) are more appropriate to HDL descriptions, while other portions (speed-critical logic) are more appropriate for schematic entry. The QuickLogic Quick*Works* toolkit uses schematic tools that can integrate Verilog or VHDL descriptions as blocks within a top-level hierarchical schematic.

Also, a design environment for PCI designs needs to provide a simulator and simulation interface that allows the designer a great deal of flexibility with vector entry and result analysis. PCI interface simulation requires modeling inputs from two different busses, each with read and write capability. The Quick*Works* toolkit meets this complex simulation need by including the Silos III simulation tool with it's design environment. This simulator allows for fast simulation times, and support of the industry standard simulation language - Verilog. Also QuickLogic provides application notes that explain the process of multi-chip simulation, should you decide to use simulate the interaction between multiple devices.

Finally, FPGA tools for PCI design should contain a compilation (placement and routing) tool which is fully automatic. Due to the simple and abundant routing architecture of the QuickLogic pASIC 1 series of devices, QuickLogic's compilation tools are always fully automatic, even when design specifications require that 100% of all pins and logic within the device are used, and the pinout has been fixed to meet PCI recommended pinouts. And if you want to use more advanced features, the QuickLogic place and route environment (SpDE) contains a user interface for visually inspecting the final chip design, determining accurate delays with the Path Analyzer, and making timing improvements with the Timing-Driven Placer.

PCI Design Using the QL24X32B FPGA from QuickLogic

The remainder of this paper describes a complete PCI interface implemented fully within a portion of a QuickLogic QL24X32B (8000 gate) FPGA. There is additional logic available on the device for implementing various pieces of memory control or interface logic that may be necessary for a complete PCI-bridge design. The user side of the interface has been designed for a generalized 32-bit device with a typical READY and READ/WRITE-strobe handshake sequence; 24 bits of user device address have also been provided. The large logic and pinout capabilities of the QL24X32B device are key to providing the necessary interface functionality in a single FPGA device. In addition, the extremely fast I/O pads and internal logic can accommodate the stringent system timing requirements of the 33 MHz PCI bus.

The design implements a PCI-compliant interface that utilizes the PCI burst transfer mode for highest data throughput. All required PCI Configuration Space registers have been implemented in a highly modular structure; readers may simply modify the necessary fixed-value registers to contain the vendor, device, and revision identification for a specific product.

While portions of this paper may appear to only address specific areas of the PCI interface, the general design concept described may be applied to a variety of applications for various processors and peripherals. The design files and schematics are available from QuickLogic and can be easily modified to your particular needs.

System Overview

Figure 1 below indicates a typical PCI system topology. The CPU is coupled to the PCI bus via the indicated adapter. The figure indicates that a high-speed disk and video controller reside on the PCI bus as well as a user device, interfaced by the QuickLogic FPGA device described in this paper.

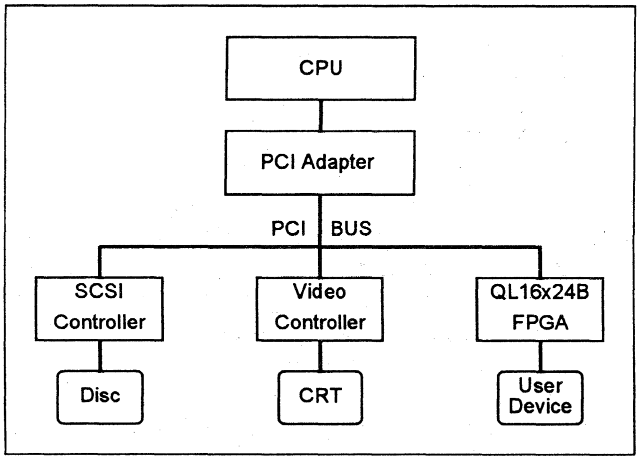


Figure 1 : System Topology

PCI Bus Overview

The PCI Local Bus is a high-performance 32-bit (or 64-bit) bus with multiplexed address and data lines. It is designed to interconnect high-performance processors and peripheral controllers for a wide variety of next-generation computer systems.

The following is a brief overview of the PCI Local Bus. For a complete, detailed description the reader should consult the PCI Local Bus Specification, distributed by the PCI Special Interest Group.[1]

Address Space

PCI defines three address spaces: Configuration Space, defined to support PCI hardware configuration, and Memory and I/O Spaces, which are used by the attached devices for actual data transfer. Address decoding on PCI is distributed, wherein each device is responsible for decoding its own address. PCI supports two address decoding schemes: *positive decoding,* where a device looks for an address in its assigned range, and *subtractive decoding,* where a single bus-coupler device accepts all addresses that were not positively decoded by some other device. With either scheme, a device indicates that it has decoded its address and thus claims the transaction by asserting the tristate signal DEVSEL#.

Configuration Space

PCI provides for totally software-driven initialization and configuration via the Configuration Space. PCI devices are required to allocate 256 bytes of configuration registers for this purpose. (NOTE: not all of these bytes need be implemented in physical logic if a read-value of zero can be generated.) Accesses to the Configuration Space require external address decoding via the IDSEL control pin, which functions as a unique "chip-select" for each device.

Data Transfer

The basic bus transfer mechanism on PCI is a burst transaction, composed of an address phase and one or more data phases. The first

[1] PCI Special Interest Group
M/S HF3-15A
5200 N.E. Elam Young Parkway
Hillsboro, Oregon 97124-6497
(503) 696-2000

clock cycle on which the FRAME# control signal is asserted establishes the address phase of the transaction. FRAME# will remain active to indicate each data phase to follow, and will deassert at the start of the final burst data phase. Handshake signals for each data phase include IRDY# (Initiator Ready, indicating the Bus Master is supplying write-data or is ready for read-data) and TRDY# (Target Ready, indicating that the target device has accepted the write-data or is supplying the read-data).

Devices connected to PCI implement Bus Commands, which indicate the type of transaction that the Bus Master is requesting. Bus Commands are encoded in the 4-bit CBE# lines during the address phase of Bus transactions. During the data phases the CBE# lines contain byte-enables for the word transfers.

Byte lane swapping is not done on PCI since all PCI compliant devices must connect to all 32 address/data bits for address decode purposes. This means that bytes will always appear in their natural byte lane, based upon byte address. In addition, PCI does not support automatic bus sizing for 8- or 16-bit transfers: the byte-enables alone are used to determine which word bytes carry meaningful data.

The maximum transfer rate of the PCI is one 32-bit word every 30ns, or 132 MB/sec. Basic timing diagrams for PCI write and read transactions are shown in Figures 2 and 3.

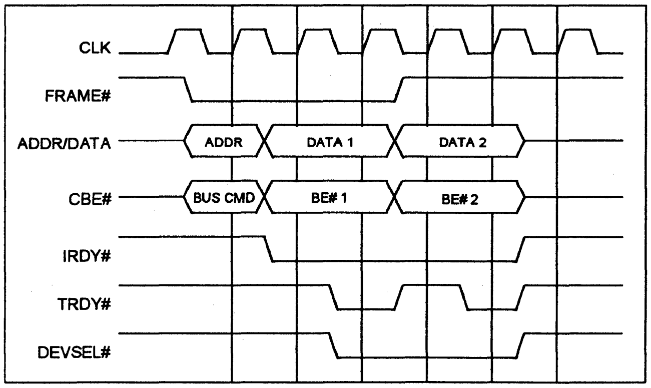


Figure 2: PCI Write Transaction

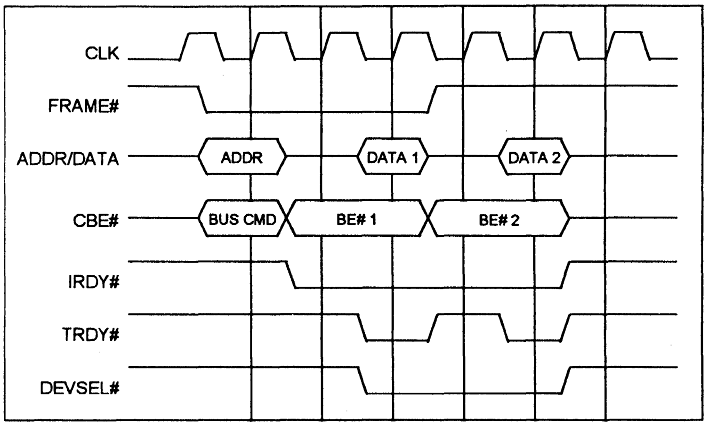


Figure 3: PCI Read Transaction

Design Objectives

The design presented in this paper accomplishes the following objectives:

- Implements a single QL24X32B FPGA device for PCI interface to general 32-bit user device
- Makes efficient use of QuickLogic FPGA architectural advantages

External Interface Signals

Figure 4 below indicates the external signals between the FPGA and the PCI Bus and user device. The function of each signal will now be described. Note that only the PCI signals that are relevant to this application are presented.

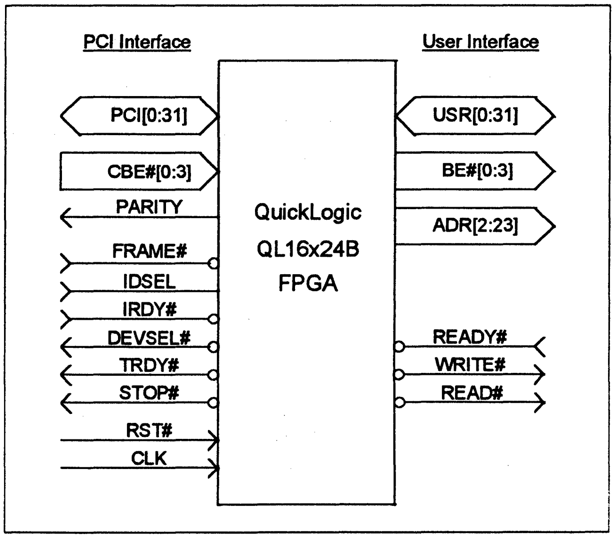


Figure 4: QL24X32B External Interface

PCI Interface Signals

PCI[0:31]	Address / Data Bus
CBE#[0:3]	Bus Command / Byte Enables
PARITY	Even parity (PCI[0:31] & CBE[0:3])
FRAME#	Data Frame (start and duration)
IDSEL	Chip select during Configuration
IRDY#	Bus master ready to transfer data
RST#	Master reset signal
CLK	Master clock signal
DEVSEL#	FPGA has decoded its address
TRDY#	FPGA ready to transfer data
STOP#	FPGA requests (burst) transfer stop

The PARITY signal is bi-directional; parity generation is required by all PCI devices but its detection is not. The PARITY signal is active one cycle after its corresponding data transfer to allow pipeline parity-generation and detection to be implemented. Parity detection has been omitted in this design for the purposes of simplicity.

The three PCI control signals driven by the FPGA (DEVSEL#, TRDY#, and STOP#) are defined in the PCI specification to be Sustained Tri-State, meaning that after being driven low (active), the FPGA must drive the signal high (inactive) for at least one clock time before letting it float.

User Interface Signals

USR[0:31]	Bi-directional user data bus
BE#[0:3]	Byte enables (write and read)
ADR[2:23]	User address bus
READY#	User device ready to transfer data
WRITE#	User write strobe
READ#	User read strobe

Note that this design assumes a separate WRITE# and READ# strobe; a simple alternative design might use a common STROBE# signal accompanied by a R/W# indicator.

The PCI interface defines transactions for memory and I/O address spaces. In the I/O address space, all address lines are valid; in the memory address space, address bits [1:0] contain burst direction information and should not be considered valid address information. This design describes a memory-space design and thus address signals ADR[0:1] are not needed by the user device.

The design assumes that the user device will be running synchronously to the PCI Bus Master Clock. However, connecting the user device and the FPGA to the PCI CLK signal would exceed the allowable fan-in capacity of one CMOS load. Thus, the system level design should utilize a CMOS Phase-Locked-Loop type clock buffer to provide zero-skew copies of the PCI master clock signal to both the FPGA and the user device.

PCI Bus Commands

During the address phase of a PCI Bus transaction the CBE#[0:3] lines contain the Bus Command, indicating the type of transaction that will occur. The defined Bus Commands are indicated in Table 1 below:

Table 1: PCI Bus Commands

CBE#[0:3]	Command Type
0000	Interrupt Acknowledge
0001	Special Cycle
0010	I/O Read
0011	I/O Write
0100	Reserved
0101	Reserved
0110	Memory Read*
0111	Memory Write*
1000	Reserved
1001	Reserved
1010	Configuration Read*
1011	Configuration Write*
1100	Memory Read Multiple*
1101	Dual Address Cycle
1110	Memory Read Line*
1111	Memory Write / Invalidate*

The Bus Commands marked with an asterisk are utilized in this design. If the reader wishes to map the device into the PCI I/O address space, a schematic module may be simply modified to decode the appropriate Bus Command transactions.

PCI Configuration Space

The PCI Configuration Space is divided into a predefined header region (64 bytes) and a device dependent region (192 bytes). A PCI-compliant device is not required to implement all of the registers; all unimplemented registers, however, must return a value of zero when read.

Of the seven indicated implemented header registers, four deal with device identification, and thus are read-only. The reader will simply modify the skeleton form of these registers provided in the design to uniquely identify the target device. The following briefly describes the contents of these registers; more detailed information can be referenced in the PCI Local Bus Specification.

Vendor ID Device manufacturer, from PCI SIG

Device ID Device type, from the vendor

Revision ID Device revision, from the vendor

Class Code Generic device function. (See detailed specification.)

The remaining three registers have read-write capability and provide the following functionality:

Command Controls the device's ability to respond to PCI cycles

Status Records status information for PCI bus-related events

Base Specifies the base address of the device (assigned by the O/S)

The Command register is cleared at power-on reset to logically disconnect the device from all PCI transactions except Configuration Space reads and writes. This design implements Command Register Bits 0 and 1 which control the device's response to I/O and memory space accesses. All other Command register bits have an implicit value of zero.

Bits [10:9] of the Status register indicate the speed at which the device is capable of decoding its own PCI address. The PCI specification offers codings for Fast, Medium, and Slow response: this design utilizes the Medium decode speed. (Fast speed is not obtainable in FPGA technology using synchronous design practices.)

The Base Register is also used by the operating system to determine the Memory or I/O space requirements of a device. In this design, the user device is assumed to have a 16-MB address space (24 bits); thus the upper 8-bits of the Base Register will be implemented with read-write capability and the lower 24-bits will always return a read value of zero. If the reader wishes to expand or contract the address space allocated to their device, the size of the Base Register and its associated address comparator can be adjusted accordingly.

PCI 64-BIT DESIGN USING LSI LOGIC'S COREWARE® METHODOLOGY

Jose A Valdes
LSI Logic Corp.
1551 McCarthy Blvd.
Milpitas,CA 95035

ABSTRACT

The Peripheral Component Interconnect (PCI) is emerging as the bus of choice across an unprecedented number of different platforms. Designing a PCI based system, however, involves a myriad of technical details dealing with I/O buffers, bus protocols, timing, device drivers, and compliance that present significant technical challenges.

This paper describes how a set of 64-bit PCI design blocks, the PCI-64 FlexCore™ Architecture, and other cores in LSI Logic's CoreWare® Libraries can substantially shorten the time needed to design and produce PCI-based ASICs.

PCI-64 FLEXCORE BUILDING BLOCKS

The PCI-64 FlexCore Architecture is a set of tightly integrated, ASIC building blocks or modules that bridge a PCI bus to an "on chip" Host interface. Combined with a User Logic block, the PCI-64 FlexCore Architecture can be used with external logic to produce a single chip PCI Host Bridge or Adapter solution. The diagram below shows a top level view of the individual functional blocks.

LSI LOGIC'S COREWARE METHODOLOGY

CoreWare is LSI Logic's proprietary approach to creating custom, single-chip systems with more complex and higher level logic blocks as opposed to standard gate arrays.

CoreWare represents a new paradigm for system designers by offering them the ability to utilize applications-optimized engineering.

With CoreWare, designers can focus on the specific application requirements necessary to make their product competitive in the market, and then quickly produce silicon optimized for that functionality and performance. Complex systems-on-a-chip can be fabricated with unprecedented time-to-market and low cost relative to alternative approaches such as gate arrays or full custom designs.

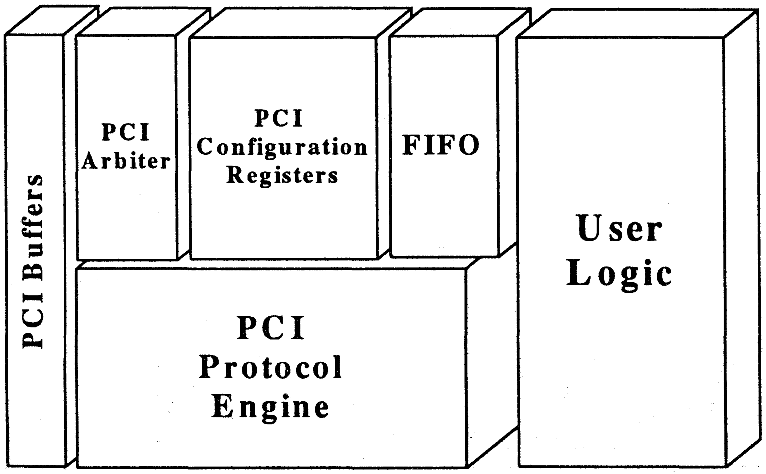


Fig. 1 PCI-64 FlexCore™ ASIC Building Blocks

WHY USE A CORE FOR PCI?

With a PCI core, the design team can focus on creating unique intellectual property, not re-engineering industry standards. By having PCI working at the start of a design cycle, high level system trade-offs are clear sooner and design dead-ends are more readily avoided. Also with a working core generating real cycles, the PCI protocol learning curve is sharply reduced if not altogether eliminated.

PCI compliance is another area where a core can save large amounts of time and resources. With the PCI-64 FlexCore Architecture, the component electrical, configuration, and protocol check lists are already complete.

Finally, reusability is a major benefit of using a core; the same core can be used in many different applications.

ARCHITECTURE

The PCI-64 FlexCore Architecture is partitioned into five high level building blocks as shown in Figure 1. These functional blocks consist of a PCI Protocol Engine, PCI Arbitration Unit, PCI Configuration Registers, FIFOs, and a set of PCI compliant I/O buffers.

PCI Protocol Engine

Logically, the PCI Protocol Engine is partitioned into six functional modules. As shown in Figure 2, these consist of muxed PCI Master and Slave modules, an Address/Data Path module, an Arbiter, a FIFO Controller, and a Host Interface Controller.

As its name implies, the PCI Protocol Engine is the workhorse of the PCI-64 FlexCore Architecture. It handles the bridging function between the PCI bus and the Host interface. The PCI side of the bridge provides all signaling for the PCI Bus and Configuration Space; the Host side of the bridge consists of the Host Interface Controller, Bridge Arbitration unit, and the FIFO controller.

The FlexCore Architecture can operate either as a master or a slave on the PCI bus. PCI Bus master cycles are initiated by the PCI Master module in response to a cycle passed by the Host Interface Controller. These cycles originate in the User Logic Block. Cycles directed towards the core on the PCI bus are responded to by the Slave module.

The Bridge Arbiter unit plays the role of traffic cop by monitoring control signals and arbitrating access to the FIFOs from both sides of the bridge.

The Address/Data Path routes 32 -bits of address and 64 or 32 -bits of data to and from the PCI Master, PCI Slave, FIFO Controller, FIFOs, Host Interface, and Configuration Registers.

The FIFO Controller generates control signals for the FIFO buffer block. Three

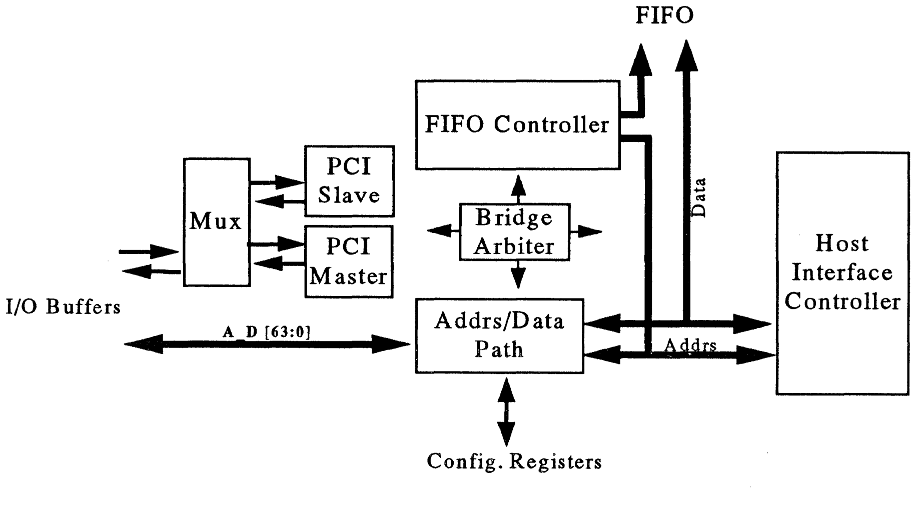


Fig. 2 PCI Protocol Engine Block

external control bits provide for different depth FIFOs.

The Host Interface Controller implements a lucid non-multiplexed Host Interface. Interfacing external logic is straightforward with an address strobe/data ready style handshake with both master and slave cycles.

The controller also takes advantage of the full capabilities of PCI with its ability to perform zero wait state bursting.

PCI Arbiter Unit

The PCI Bus Arbiter is fully compatible with PCI's access-based central arbitration scheme. It uses a request-grant type of handshake to grant mastership on the PCI bus. Support is provided for up to five masters on a programmable priority basis with parking ability.

Since some adapter card implementations do not need an arbiter unit, the PCI Bus Arbiter is optional.

PCI Configuration Block

The Configuration block implements commands that access the PCI Configuration space and provides address decode for PCI bus transactions. Within this block are a number of registers whose contents are unique for a given implementation, this module is therefore left as a soft layout. and VHDL source code is available for custom implementations.

The Configuration block can define up to six resources that may be used in each of four Host Interface devices. The resources may be I/O resources or memory resources, where the size of each resource is programmable. The memory resources also have a relocation attribute.

Additional user definable options in the standard implementation are shown below.

The Global options:

- Vendor Identification
- FIFO Size

Local Device options:

- Device Identification
- Revision number
- Base class number
- Subclass number
- Program interface
- IRQ pin mapping
- Resource register settings

Resource Register options:

- Enable/disable resource
- Memory or I/O select
- ROM resource select
- Resource size select
- Memory relocation attributes

FIFOs

The FIFO is assumed to be a memory block with separate read and write ports. The FIFO can be 4, 8, 16, 32, or 64 double words deep to accommodate different performance requirements. The size of the FIFO selected determines the core's maximum sustainable bandwidth.

PCI Buffers

The PCI specification very precisely defines the electrical characteristics and constraints of components, systems, and expansions boards on a PCI bus. Producing a compliant I/O buffer requires extensive modeling, simulation, and characterization. The task is complicated further since PCI specifies both 3.3 Volt and 5 Volt signaling environments.

LSI Logic has a number of I/O Buffers in its ASIC cell libraries that are fully PCI compliant. The RBD12PCI and RBD12PCIF buffers (in 500K Process Technology), for example, supports 3.3 Volt and 5 Volt signal environments.

DESIGN METHODOLOGY

Object Oriented VHDL

The PCI-64 FlexCore Architecture was designed using a VHDL top down design methodology encompassing "Object Oriented" concepts. By abstracting the core into separate functional elements, many different design manifestations are possible without changes to overall design integrity.

Design Partitioning

The top level modules are partitioned into either fixed layout "hard macros" or user modifiable netlists. The PCI Protocol Engine and PCI Arbiter blocks, for example, are "hard macro" blocks since they contain critical timing paths which should not be altered.

The PCI Configuration registers are "soft macros" since register settings for things like Vendor ID and Device ID must be different in each design.

The FIFO memory block has been isolated from the rest of the core so different memories can be used to vary sustained data bandwidth rates.

Third Party Simulator Support

The PCI-64 FlexCore Architecture can be used in many different system verification environments. Though primarily targeted towards VHDL 1076 compliant simulators, Verilog-based design environments are supported if the simulator supports co-simulation or has VHDL model import facilities.

Support for other types of environments is accomplished with LSI Logic's C-MDE tool set. With C-MDE a netlist can be translated and back annotated into virtually any simulation environment.

FULL PCI 2.1 COMPLIANCE

PCI compliance can be a drain on engineering resources. The PCI-64 FlexCore Architecture has been thoroughly tested for PCI 2.1 specification compliance. With the Host Interface decoupled from the PCI side, the designer need only be concerned with connecting to a more forgiving on-chip Host Interface.

LSI Logic's CoreWare library qualification requirements in many ways are more stringent than those of PCI. Rigid design methodology, a complete system verification environment, post and pre-layout timing analysis, and back-annotation into the SVE are done to check functionality.

Once a core has been successfully simulated and layed out, prototypes are generated, tested, and characterized on both a chip tester and at the system board level to assure full silicon functionality.

INTEGRATING OTHER COREWARE COMPONENTS

The PCI-64 FlexCore Architecture combined with LSI Logic's comprehensive library of functional cores allows the implementation of PCI functionality to a significant number of industry standards.

The CoreWare library includes a wide variety of leading microprocessors, networking, and high speed interface elements, including RISC microprocessor, Ethernet controllers, ATM networking cores, DSP, Fibre Channel, MPEG2, JPEG and DigiCipher II video, and Dolby AC-3 audio cores.

CONCLUSIONS

The PCI-64 FlexCore Architecture offers a new approach to system design that allows the system designer to test many designs in a CAD environment at the start of the silicon design.

By allowing selection of such key features as Bus Arbitration, FIFO depth, and device resources, the core provides the flexibility required for system designers to meet a broad range of price performance goals.

By isolating the complex electrical and timing specifications of the PCI bus standard, the PCI-64 FlexCore Architecture enables engineers to focus resources on adding value to PCI Host and Adapter designs.

On-Chip SRAM Aids PCI Design

David Ridgeway
Market Segment Applications Manager
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3450
(408) 879-4671/371-6648 (fax)
davidr@xilinx.com

Equipment manufacturers developing PCI-based systems face the challenge of designing products quickly while, at the same time, addressing the requirements of the new and evolving local bus standard. An ASIC can offer a cost-effective solution here, but the long development time suggests that it is an unacceptable option. FPGAs with on-chip SRAM provide a better and faster approach to designing PCI local bus interfaces. in addition to in-system reconfigurability in computer systems, on-chip SRAM can be used to implement scalable, high-performance FIFO buffers to maximize the bandwidth offered by PCI.

TRENDS IN MOTHERBOARD DESIGN

Jeffrey Lee
First International Computer, Inc.
980A Mission Court
Fremont, CA 94539

ABSTRACT

Since the first introduction of the affordable personal computer, the requirement for faster and better performance has increased. Fueled by consumer and technological demands, manufacturers have been forced to develop products to keep abreast of and ahead of these demands or risk extinction.

Current trends in motherboard development have been based on:

- Advances in CPU performance
- Larger and faster Cache and DRAM
- Enhancements in I/O buses
- "Plug & Play"
- Advanced networking requirements

CPU PERFORMANCE

Intel has been one of the primary driving forces behind CPU developments, spearheaded by the Pentium Processor. AMD's K5 and Cyrix's M1 further fuel the race toward higher performing CPUs and the need for advanced system logic designs.

CPU performance has grown at an average of 50% per year. As performance increases, so too does the demand for hardware capable of fully utilizing CPU interface designs.

CACHE & DRAM

Coinciding with advancements in CPU performance is the demand for more system memory. Today's hardware and software applications already require more memory than ever before. Tomorrow's demands must equally be considered.

System logic designs for the Pentium already support up to 2MB cache, while future designs will support synchronous pipelined 3.3v cache.

New developments in DRAM are also creating changes. EDO DRAM, synchronous DRAM, EDRAM, RAMBUS, and 3.3v DRAM are all factors to be analyzed in new motherboard designs as system logic developments rise to meet the challenges.

I/O PERFORMANCE

With PCI's 32-bit data bus, multiple peripheral components and add-on cards can reach a bandwidth of 132MB/sec. Low access latency allows faster access times, eliminating the need for additional memory on add-on cards and thus lowering costs. Additional performance is reached through PCI's bus mastering support.

Increased developments in graphics, multimedia, disk drives, and networking further highlight the need to improve the I/O bottleneck.

"PLUG & PLAY"

PCI's "Plug and Play" support is a major issue for board manufacturers. Today's average user is not technologically sophisticated, and demands for user friendliness is on the rise. With a growing number of users eager to continuously upgrade, PCI's built-in configuration registers and software allow users to add new hardware effortlessly, bringing the age of computing closer to the consumer.

NETWORKING

While networking has filtered down to the SOHO marketplace, there is an increasing demand for high performance enterprise networking. Advancements in hard disk drives, telecommunications, and network hardware further solidifies the need for high performance bus architecture. PCI architecture allows peripheral functions to take advantage of the CPU's processing power without depending solely on the CPU itself. PCI's higher bandwidth, low access latency, and bus mastering can increase network performance

SUMMARY

As CPU and system logic designers continue to improve their products, PCI architecture offers a cost effective solution to board real-estate, with fewer components needed to populate the board. This further allows manufacturers a quicker development time, and faster time to market.

THE OVERVIEW OF PCI BASED MOTHERBOARDS

Leonard Tsai
Director of Engineering
Elitegroup Computer Systems, Inc.
45225 Northport Court
Fremont, California 94538
Ph. (510) 226-7333 Fax (510) 226-7350

Abstract

Since the announcement of PCI specification, there has been several generation of implementations for the mother board chip sets. This paper examines their architecture, implementation and provide comparison information from actual commerical products. We hope in this way, reader can obtain better understanding of the decision and trade off made during the design phase.

1. The Historical factor

When PCI bus specification was first announced, the market already accepted VL bus definition for low cost and its simplicity. Defined by VESA (Video Equipment Standard Association) work group to support higher performance graphic display required by Microsoft Windows 3.1, VL bus is simple and glue to the 486 processor bus protocol. During the definition stage of the VL bus, Pentium Processor was still un-announced. Although many of the participants in the VL bus work group have knowledge of how the upcoming processor will look alike, but none of them can discuss it in public. Therefore, the VL bus was very easy to adapt in 486 type design but lack the proper support for Pentium functionality.

During that time, Intel's 82420 (codename Saturn) was the only shippng chipse. Many companies design motherboards based on the same reference design guide from Intel. Unfortunately, chip set bugs and changes cause multiple revisions of product within a short period of time. After piles of errata and correction with external GAL fixes, the market abandon Saturn based 486 PCI products eventually.

In the same time Pentium processors was introduced. Several companies came out with a quick design and come out Pentium mother board with VL bus. However, armed with astronomical budget, Intel lunch series of marketing advertisement and lead the major PC marketing companies such as Gateway 2000 to Pentium PCI market, the demand for Pentium/VL motherboard disappear very soon. At that time, the only available Pentium PCI based chipset is from Intel (82430, codename Mercury). Opti, Inc. later came out with with 596/597 chip set to ease the dominance from Intel.

It is not until Intel start shipping P54C processor that other chip set supplier finally catch up. From Taiwan UMC, the 881 chipset (for 486) and 891 chipset (for Pentium) are good low cost, PCI 2.0 compliant solution. Opti, SiS, VLSI, Acer Labs and Via Technologies, all ship their PCI chipset in 1994. Today, there are more and more PCI mother board chip set coming out and make designer job much easier than before.

2. Architecture

We will skip the 82420 Staurn chip set are it is already obsolete. In Figure 2 and Figure 3 we demonstrate the architecture differences of 80430LX (Mercury) and 80430NX (Neptune).

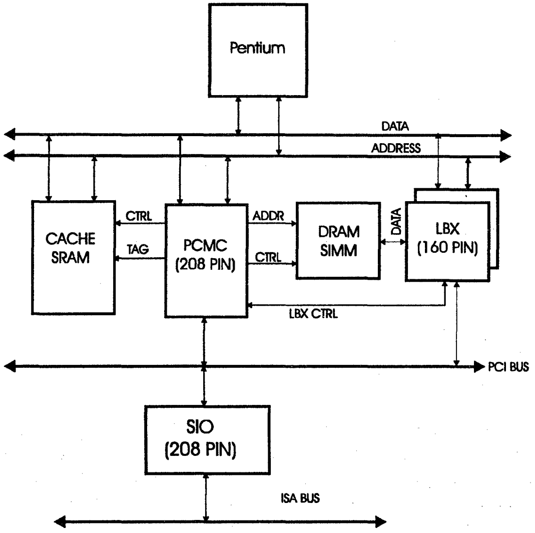


Fig 2. 80430LX (Mercury)

We notice there are only minor changes but the basic architecture is very similar between Intel Mercury and Neptune chip sets. One major differences between these two chip sets is that Neptune support dual P54C processor which requires 3.3V interface in PCMC and LBX.

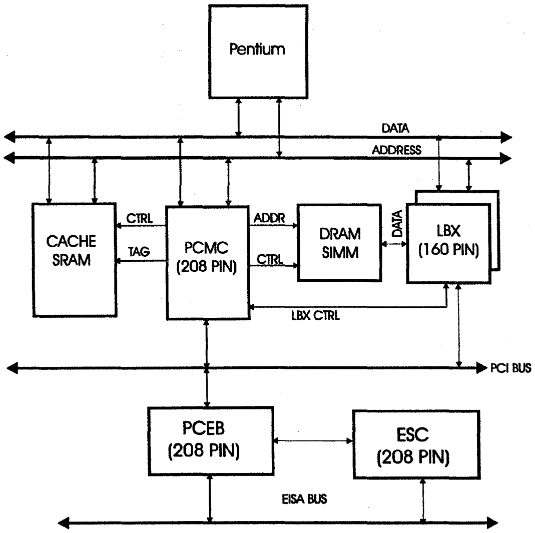


Fig 3. 80430NX (Neptune)

The new generation PCI chip are more performance tuned. From Fig. 4, we can see that VLSI 590 chip set has shrink the datapath buffer size and use PCI bus to pass the command and status between the PSC (Processor/System Controller) and Data buffer. Notice the 32 bit address/data path is interleave to reduce noise and increase efficient during byte or word transfer.

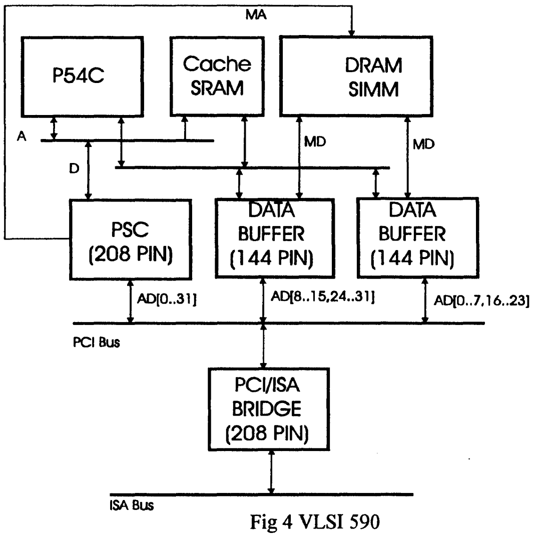


Fig 4 VLSI 590

Intel new generation Triton chip set also use this type of approach. In Fig. 5, we can see that the LBX in original design has been shrink into TDP (Triton Data Path) in 100 pin PQFP to reduce cost. However, the command/status bus still exist between TSC (Triton System Controller) and TDP. PLINK (16 bit) bus become the main data path between host CPU and PCI bus.

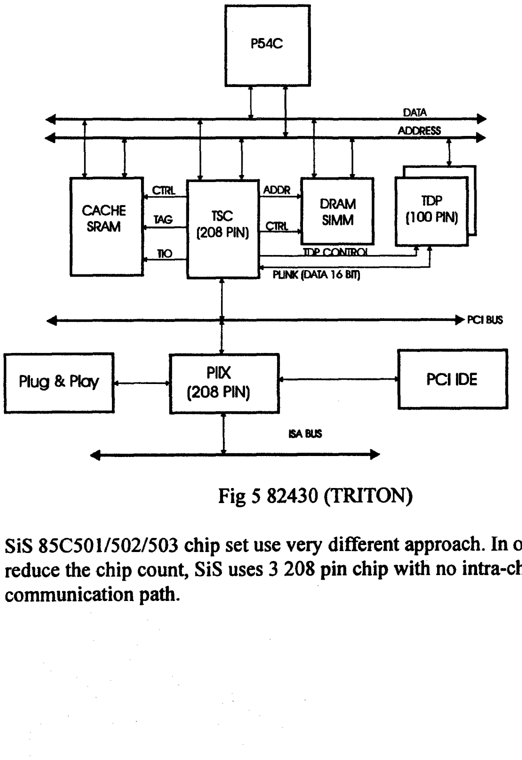


Fig 5 82430 (TRITON)

SiS 85C501/502/503 chip set use very different approach. In order to reduce the chip count, SiS uses 3 208 pin chip with no intra-chip communication path.

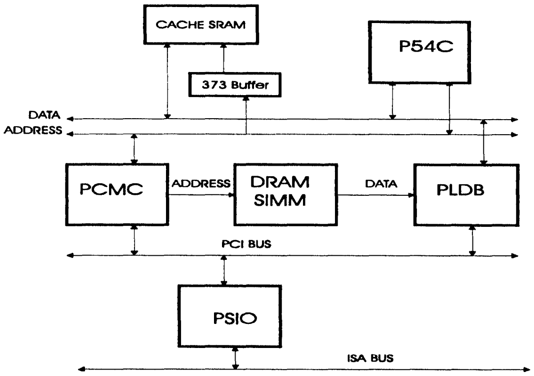


Fig 6. SiS 85C50x

One of the major challenge in chip set design is pin count. At current stage, SiS 3 chip solution seems to be the most compact design. Next generation PCI chipset will use Ball Grid Array (BGA) with 300+ pin count and futher reduce the chip count down to 2. At that time, because the data path is integrated into the controller, with help of internal buffer, higher performance PCI design will be possible without major change to architecture.

3. Performance and data buffer

In the first generation PCI chip set from Intel, we notice the extensive use of buffers and FIFO. There are several levels of these buffers help to minimize the overhead and turn around cycle penalty. Intel has implement buffers in several different key data path which help to maintain concurrency for PCI and processor bus. However, these buffers requires very complex arbitration logic and require large real estate on the silicon, they increase the cost of the chipset and also reduce the initial yield rate. Several companies make different choice and take the performance penalties. In Table 1, we can see the comparison of the buffer usage of different chip set. In Fig. 3, we can also see some raw performance number sampled from real products.

Table 1

In Figure 6 and 7, we use Intel 80430 chip set LBX (Local Bus Accelerator) and SiS 85C502 PLDB (PCI Local bus Data Buffer) as example to compare their internal buffer structure. These buffers in the PCI mother board chip set plays a very important role to keep the system from waiting each other. It also help to introduce concurrency between host bus and PCI bus.

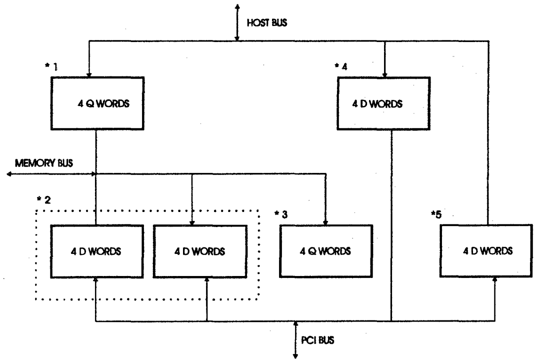


Fig. 6 LBX Buffer

Notes in Figure 6 represents :

* 1 CPU-to-Memory Post Write Buffer
* 2 PCI-to-Memory Post Write Buffer
* 3 PCI-to-Memory Read Prefetch Buffer
* 4 CPU-to-PCI Post Write Buffer
* 5 CPU-to-PCI Read Prefetch Buffer

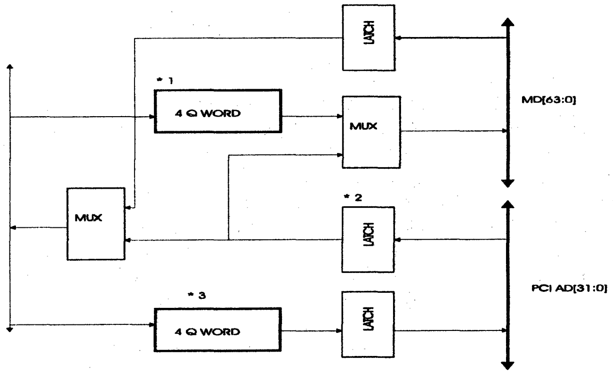


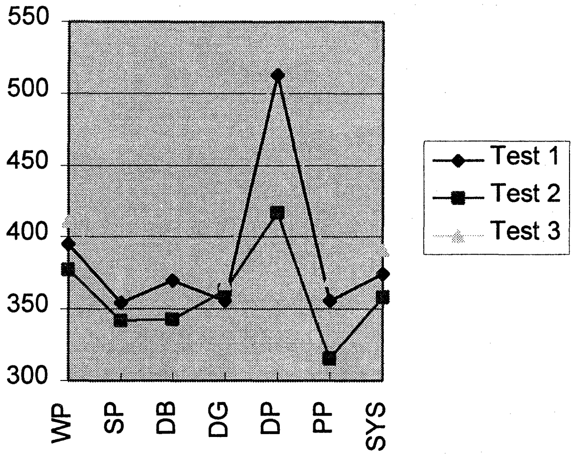
Fig. 7 PLDB Buffer

Notes in Figure 7 represents :

* 1 CPU-to-Memory Post Write Buffer
* 2 PCI-to-Memory Post Write Buffer
* 3 CPU-to-PCI Post Write Buffer

We can see the difference between LBX and PLDB is that PLDB did not incldue prefetch buffer. These prefetch buffer can help to reduce the wait cycle and improve the system performance. Also, there are no PCI to CPU buffer which will also impact system performance when PCI bus master try to access the system memory.

In Figure 8, we chart real life several benchmark result to demonstrate the impact of these buffers.



Test #1 is with all buffer turn on, Test #2 is with PCI post write buffer turn off and Test 3 is with Host post write buffer turn off. Test is performed under BapCo 93 (SysMrk93). Legend : WP - Word Processing, SP - Spread Sheet, DB - Data Base, DG - Desktop Graphic, DP - Desktop Publishing, PP - Presentation.

Fig. 8

4. Layout Issues

The PCI mother board layout has cause many engineer spend countless overtime and retries to try to get the best placement on the limited PCB space. Because PCI specification defines all peripheral components are mounted at reverse side of ISA card, a share slot scheme was defined. The placement of share slot has cause some challenge to engineers to come out with different placement than traditional mother board 3 years ago. We will examine two major types of mother boards; standard Baby AT and LPX/LPM boards for slim line. There are several other form factors that uses full size AT or passive backplane for special industrial application. However, we will focus on the most popular types only.

4.1 The movement for on board I/O

One of the major movement in PCI based mother board is that almost all of the new PCI based mother board has Super I/O chip on board. This is the new trend in the industry. There are several good reason behind this movement. First, the PCI specification defines only 1 load per slot, and it limits traditional COMBO type of I/O card design unless a very expansive PCI to PCI bridge chip is used to meet the compliance. Second, by definition, PCI bus is Plug and Play ready and by putting the Super I/O chip on board, the designer can have full control in BIOS setting which allow easier transition to full Plug and Play in future. Third, the new emerging Enhanced IDE specification from ATA committee really benefit from the high speed PCI bus transfer rate. Because the PCI specification did not include the INT 14 which is used by the primary IDE channel, the PCI IDE add on card will need a paddle card to get INT 14 signal from ISA slot side. This will in turn use 2 slot instead of 1. And last, by putting the PCI storage controller on board, the mother board designer can fine turn the parameter and boost the transfer rate and give better performance.

The on board I/O today normally includes a PCI IDE chip and a Super I/O chip which provide the basic floppy controller, 2 FIFO serial port and 1 ECP/EPP printer port. However, this type of design does has 1 draw back that requires mother board manufacturer to extend the product test and support to peripheral devices that are attached to these I/O ports.

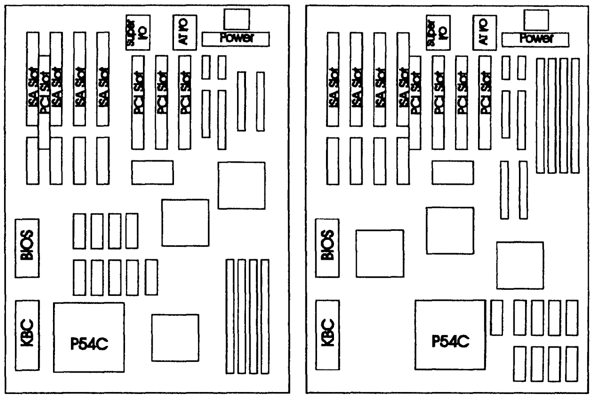


Figure 9.

4.2 Baby AT layout examples and issues

The original Baby AT formal factor host 8 ISA slots. Under the same formal factor and constrain, the PCI mother board designer has to divide the slot space between PCI and ISA (or EISA). Normal configuration today includes 4 PCI and 4 ISA or 3 PCI and 5 ISA slots. There are 2 types of the layouts that are widely adapted. (See Fig. 9) In Fig 8, the black lines marks normally trace routing on these placements.

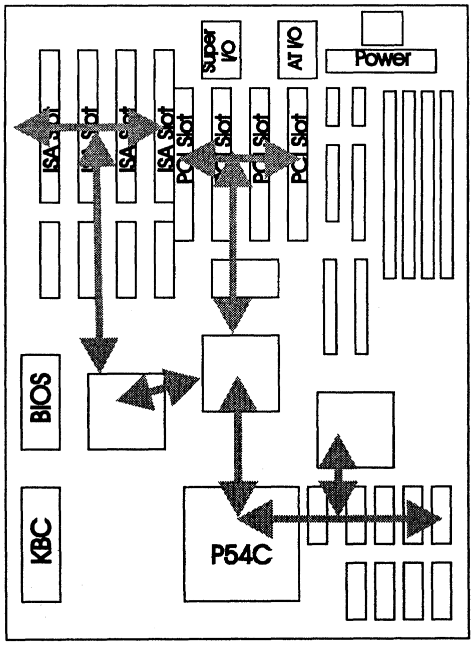


Figure 10.

The general problem on layout #1 in Figure 9 is the component placement. We notice that the memory is located near the upper right corner which blocks any trace between the MC and I/O chip block. The Super I/O and I/O interface chip are located at the read of the PCI slots and still meets the mechanical requirements of PCI specification. However, the I/O connector placement become a bigger issue that if they are placed next to the memory SIMM slot, then the memory slots is limited to 4 SIMMs only. If they are placed near the power connector at top, then memory SIMMs has to move down and may interfere with the floppy disk drive in the chassis.

Because these limitation, the PCI bus controller or data buffer chip normally is place right under PCI slots. This will provide a very efficient layout between the MC and the PCI controller. The MC is placed near the center of the CPU and the cache memory and the net tree is flat (linear) using minimum distance. Sometimes, the design engineer need to manually adjust the net list sequence to obtain the desired result before the layout engineer start to do the placement.

The layout #2 in Figure 10 also have placement problems. As the memory SIMMs are located in the lower right corner, it pushes the MC and CPU to left of the board which in turns blocks the ISA slot for full size card. Because the space limitation, the cache memory get push sit between CPU and MC which place right under the PCI slots. Since the traffic between CPU and Cache memory is very high, the simultaneous switching noise become noticeable from the PCI slot side. Extra filters and bypass capacitors are needed but normally translate into cost and routing constrain.

The PCI mother board normally take several iterations before the balance of cost, manufacturablity, and reliability can be reached. The layout process normally is break into two phases. The trivial traces between the buses are first lay down to achieve regularity (for both PCI and ISA). Then the connection between bus controller and on board PCI device is done to obtain minimum distance layout. Last, the controller is connected to the PCI slots. All the standard PCB layout rules for high speed logic applies and to minimize the clock skew between slots, it is recommended that the clock trace are placed manually with special treatments.

4.3 LPX/LPM layout issues

LPX PCI board design is particularly difficult because it involves PCI/ISA mixed riser card. The mega connector on the mother board normally serve as the great wall on the board that blocks and trace from left side of the board to the right side. Because of the physical limitation and space constrain, there are little choice for component placement for design engineer to choose from. In most LPX form factor case, the on board I/O device must be placed at left side of the riser card slot. This is for easy cable routing consideration. In Fig 11. we can see a typical layout placement.

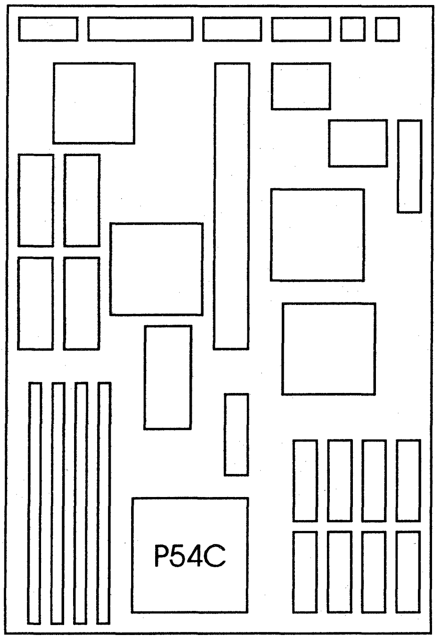


Figure 11.

In this placement, we notice that the cache memory is located at lower right corner. The memory SIMMs are located at lower left which give easy access for the user. Since the upper left corner are occupied by VGA controller and video memory, there leave very little room for the placement of chip set. One design dilemma is that for the consideration of the space, designer may choose to put the PCI IDE (or SCSI) device on the left side on the riser card connector and put the PCI controller on right side of the slot. The draw back of this implementation is that if the device plug into the PCI slots on the riser card cause any signal degradation, then the PCI IDE chip may not receive good enough signal quality and may cause intermittent problems.

One good rule of thumb is to consider a safety factor for PCI slots on the riser card. Two PCI slots on the riser may be safe if there is one or two devices on board already. Routing of the clock and some critical bus control signal is very important and designer should pay special care to them.

4.4 PCI Riser card

The PCI riser card is especially difficult to design due to the mechanical difference between ISA and PCI add on card. The same share slot issue we discussed before still apply here. In Figure 12, we can see several design that has been proved to work.

The PCI riser faces two fold of the problem; first, because the PCI component are mounted on the reverse side of the ISA add on card, the PCI slot has to be half slot lower then the ISA card. In a single sided riser card, there is no problem except the share slot issue.

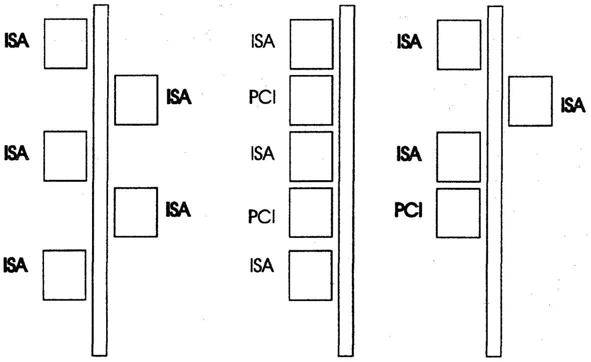


Figure 12.

In the dual side PCI/ISA mixed riser card, this post major as one side of PCI slot will conflict with the ISA slot on the other side. Special arrangement has to be made and case bracket designed to meet with this.

Signal integrity should be considered before any slot arrangement. Marketing requirement may like to place PCI slots on top of the riser card and ISA on the lower part to use 1 share slot in between. This placement will cause problem when high speed signal for PCI slots travel between ISA connectors. Interference and noise has been observed during prototype experiments when there are analog device reside on ISA slot which requires clock signal from mother board (e.g. osc, 14.318 MHz)

5. A look into Mixed PCI design and future

Several of the possible direction for the current PCI mother board design are outlined here. In next 2 or 3 years most of these functions should start to show up in market. Will they be widely accepted and design into future products is left to be seen.

5.1 3.3V PCI bus

Although 3.3 V PCI bus is already in several notebook design but desktop system still have not adapt yet. One of the main reason is 3.3V component normally cost more. The new Pentium P54C processor has help to proliferate the support on 3.3 V on mother board. But not until the 3.3 V memory price goes down will the 3.3 V become major factor.

5.2 66 MHz PCI bus

The new proposal in PCI workgroup for the extension of 66 Mhz bus specification is far more difficult for mother board design than expected. To extend the 4 PCI slots to 66 Mhz not only increase signal integraty problem but also cause EMI headaches. New architecure and deeper bus buffer will be needed to support the operation and thus increase the chip cost.

5.3 64 bit PCI bus

64 bit PCI bus has not catch up in the market but in a few year when EISA based server gradually evolve into PCI, 64 bit PCI bus may be a good choice for heavy server application. Large and wide disk raid controller can take the advantage for the wide transfer bus and provide high bandwidth. One of the potential targetfor 64 bit bus is graphic display card. However, this will rely on BGA technology and the final component cost.

5.4 PCI on PCI (Mezzanine bus)

PCI Mezzanine bus was lead by DEC as a way to expand PCI to more device general purpose bus. Later on, this was proposed as the solution for multi-drop video data path as compare to the VMC (VESA Media Channel). At current stage, most of PCI bridge design in done on multi-function add on card. Small quantity of server boards with more PCI slots design has been done in a very limited scale.

Reference

Intel 82420 PCIset Cache/Memoru Subsystem, Intel Corp.

Intel 82430 PCIset Cache/Memoru Subsystem, Intel Corp.

PCI Specification 2.0, PCI SIG

Solari, Willse. PCI Hardware and Software Architecture & Design, 1994 Annabook

PCI IN INDUSTRIAL SBC/ PASSIVE BACKPLANE APPLICATIONS

Give industrial applications the advantage that's offered by the PCl local bus.

BAO TRAN
Senior Design Engineer,
Teknor Microsystems, Inc.
616 Cure Boivin
Boisbriand, PQ. J7G 2A7 Canada

INTRODUCTION

Industrial applications now can take advantage of the speed and processing power of the latest PCs. That's because the PCI local bus supplies an effective 32- or 64 bit I/O bus, effectively eliminating bottle-necks between the system processor and high-bandwidth peripherals, such as graphics displays video interfaces, and LAN Controllers. With PCI, the system under design can exploit the latest high-speed microprocessors using a flexible, CPU-independent bus.

In the following pages, we will introduce the Industrial PCI specification and examine the unique challenges involved in designing PCI-based passive backplane and single board computer systems for industrial applications.

MOTHERBOARDS VS SINGLE BOARD COMPUTERS

Motherboard computers do a terrific job in a great variety of applications. While they are utilized primarily in desktop systems, many industrial computer setups do actually utilize motherboards. Despite the fact, however, that motherboard MTBF's are continuously improving, there is still a significant advantage to employing passive backplanes and single board computers (SBCs) in typical industrial applications.

For starters, accessing single board computers for repair or replacement is typically much easier and quicker than for motherboards. That is, a user's mean-time-to-repair is greatly reduced with passive backplane systems. In addition, motherboards are limited in the number of expansion slots, whereas passive backplanes can offer the maximum number of dedicated I/O slots for any particular application. And finally, passive backplanes and SBCs are much more flexible in terms of size and format and can be easily customized for special applications.

WHY PCI BUS IN INDUSTRIAL APPLICATIONS

Because of the continual technological advances being made, industrial system designers increasingly require high-speed I/O in their designs. Technologies such as ATM, high-speed LANs, extended 3D graphics, and so on, have resulted in system bottlenecks in ISA-based systems.

PCI has the necessary throughput designers are looking for to mate with the new technology. In addition, PCI local bus technology offers processor independence which, among other things, allows you to interface to a wide variety of CPU platforms with only one design.

AN INDUSTRIAL PCI SPECIFICATION

The PCI Industrial Computer Manufacturers Group (PICMG), is an industry marketing and technical work group setup to standardize an industrial PCI passive backplane specification. The group adopted a specification guideline proposed by charter members Teknor Microsystems, Montreal, Canada, and Trenton Systems, Duluth, Ga. That guideline helps maintain plug-and-play compatibility between CPUs and backplanes.

Known as the P996 standard, the specified CPU board is a full-size AT board. It measures 13,330 by 4.5 in., and is made up of two sets of finger connectors. The first set is a standard ISA connector, which supports the ISA-bus backplane. The second set is a modified version of the PCI bus, which supports four PCI masters on one backplane.

A board can run on both the newly designed PCI-ISA backplane or on an existing ISA-only backplane. Four clock drivers facilitate clock layout on the backplane, as well as minimize clock-skew problems when driving the expansion slots. The CPU board supports the PCI and ISA buses independently. Combining the two buses gives users the best of both worlds: High-speed devices can use the PCI bus, while the ISA bus supplies the necessary backward-compatibility with existing peripherals.

PICMG ultimately hopes to get approval of the PCI backplane specification from both the PCI SIG and the IEEE.

INDUSTRIAL PCI DESIGN CHALLENGES

Designing the PCI bus into an industrial single-board computer (SBC), however, requires careful attention to a number of design issues. The design process begins with a definition of the system under design-- an ISA- and PCI-compatible system with more than three PCI slots, and a passive backplane or motherboard.

Suppliers of SBCs needed to review the PCI standard to define the "custom bus" for the CPU board, which can support the PCI expansions slots on a backplane. As part of this approach, extra pins were added for point-signal connectors, and for the location of the passive components. These modifications support PCI's 64-bit orientation and offer backward compatibility with existing applications that don't require PCI's bandwidth *(see Figure 1).*

The PCI-ISA backplane consists of one CPU connector slot that combines an ISA connector and a 32- and 64-bit PCI connector. The ISA connector is close to the bracket end of the backplane, followed by the PCI connector, so that the CPU board also functions as an ISA backplane. Using the specified backplane layout, the ISA connectors can be added to one side of the CPU connector while the PCI connectors are on the other side. The connector layout follows both ISA and PCI standards, making it possible to fit the backplane into most current industrial enclosures. Another advantage of this layout is that the system integrates both standard PCI and ISA boards developed for desktop computers.

The CPU board is a full-size card, whose optional "drop-end" is used for peripherals I/O connectors. The remaining drop-end allows four 72-pin SIMM connectors to be used for memory-expansion purposes.

Unlike the pin assignments of the ISA connectors, the assignments of the PCI connector have been modified. This was done so that the primary bus could support up to four masters. Function pins not used for industrial applications were removed. To meet the 2-ns PCI clock-skew requirement, each dot is driven by one line from the clock driver on the CPU card. This requires an extra pin for each expansion slot..

This revision of the PCI specification clarifies certain connection schemes for SBC and passive backplanes. Such connections include pull-up resistors that support 64-bit expansion boards.

The number of slots is limited by a 10-ns propagation delay imposed by the PCI standard. Because the PCI connector on the CPU card is located below the ISA connector, 9 in. of trace length are needed to route the signals across the backplane to the three expansion slots. The remaining 4 in. can be employed on the CPU board.

The PCI bus uses the reflected wave in a constructive manner. The summation of the original signal and the reflected signal can drive low-impedance lines. If the trace lines are longer than 13 in., the reflected wave will take longer than 10 ns to return. This leaves the original signal alone, which is incapable of driving the lines, thus causing system malfunctions. The designer also must ensure that less than 2 ns of clock skew exists between any two devices. In this case, each expansion card should be driven by a separate clock from a clock driver on the CPU card. If there are PCI devices on the CPU card, a delay can be added on the clock line. The delay ensures a clock skew of no greater than 2 ns for the PCI devices and any other devices using the expansion slot.

Transmission-line effects should be considered when calculating trace delays and signal integrity. For a standard material board, a 0.2-ns/in. delay for strip line and a 0.15-ns/in. delay for microstrip line is commonly expected. Because the signal-switching is based on a reflected wave, line impedance must be controlled. The 2-ns clock skew, including the clock-driver and trace skew, requires a careful choice of clock buffers and click-trace layout. In addition, the connector selected should support high-speed signals as well as a low load on the transmission line.

A set of point-to-point signals should be added to the CPU connector for each expansion connector supported. According to the PCI standard, a set of signals such as Request, Grant, and ID Select is unique to each PCI expansion connector. Those signals should be supported by the CPU connector. The ID Select signal is a restively coupled version of one of the address signals. The CPU connector pin can be saved if these signals are routed on the backplane, so that the ID Select signal is independent of the address signals on the backplane.

The coupled Request and Grant signals support the master mode. Each PCI device possesses Request and Grant signals that interface with the PCI master controller. The number of extra pins that must be added to CPU connector depends on the number of master-enabled expansion slot supported and on the chosen PCI master controller. Four to six masters can be supported by the PCI master controller and arbiter using available chip sets.

The location of certain components is critical (on either the CPU board or backplane) for the system to run with either the 32- or 64-bit versions of PCI. According to the PCI standard, the Request-64 and Acknowledge-64 signals negotiate a 64-bit transfer between devices that support 64 bits. Selecting the proper pull-up-resistor scheme must be considered to avoid contention problems, such as a 64 bit expansion card being mounted in a 32-bit slot.

The PCI bus, which is synchronous, features signal switching based on a clock edge. The address lines can change state on every clock cycle when in burst mode. To avoid EMI problems, several techniques can be utilized, including lowering the trace impedance and running high-EMI-generated lines on the inner-layer board. Crosstalk can be avoided by leaving enough space between signals and rerouting signals that are sensitive to noise away from high-speed switching signals, such as the clock lines.

The PCI bus supports existing standard busses including ISA, EISA, Micro Channel, Multibus, and VMEbus. System designers can integrate one of these buses to support existing designs and applications. To facilitate a smooth transition from the existing technology to the PCI bus, the system controller on the CPU board should support both buses independently. All transactions between busses must be transparent to the applications. One or both buses can be omitted from the backplane.

PCI TO PCI BRIDGE

In industrial applications, it's common to use more than eight expansion slots. For such applications, a PCI-to-PCI bridge offers a solution. The bridge, when connected to one PCI bus, called the primary bus, creates an additional PCI bus, called the secondary bus. Up to four additional slots can be supported for each bridge. The bridges can be added in hierarchical order or in peer, depending on the application. The bridge also permits the PCI buses to run concurrently as the devices on one bus don't access the devices on another. Within the PICMG specification, up to four PCI to PCI bridge slots can be added together in *peer mode* to support up to 16 PCI slots on the secondary bus *(see Figure 2)*.

A FINAL NOTE

During the layout of a high speed board, trace length must be viewed as a transmission line. Similarly, crosstalk, EMI, signal integrity (overshoot and undershoot, ringing, and skew) should undergo analysis using the appropriate tools, In the design of Teknor's PCI boards, the PDQ tool was used for component placement. XTK was utilized for transmission line and crosstalk analyses.

Bao Tran, is a senior design engineer with Teknor Microsystems Inc. He holds a degree in electrical and communications engineering and is concluding a master's degree in computer engineering at the Ecole Polytechnique in Montreal, Canada.

Figure 1

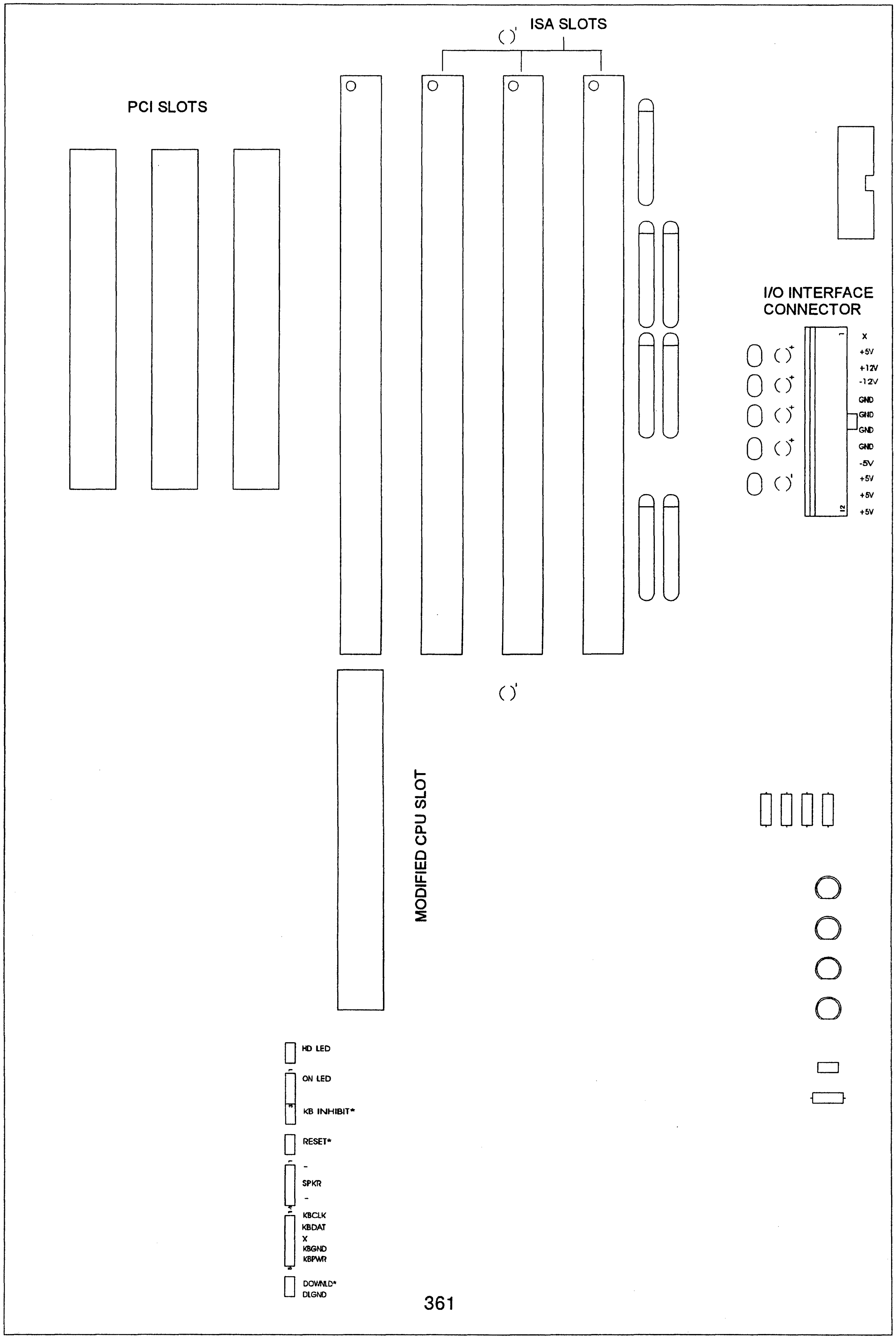
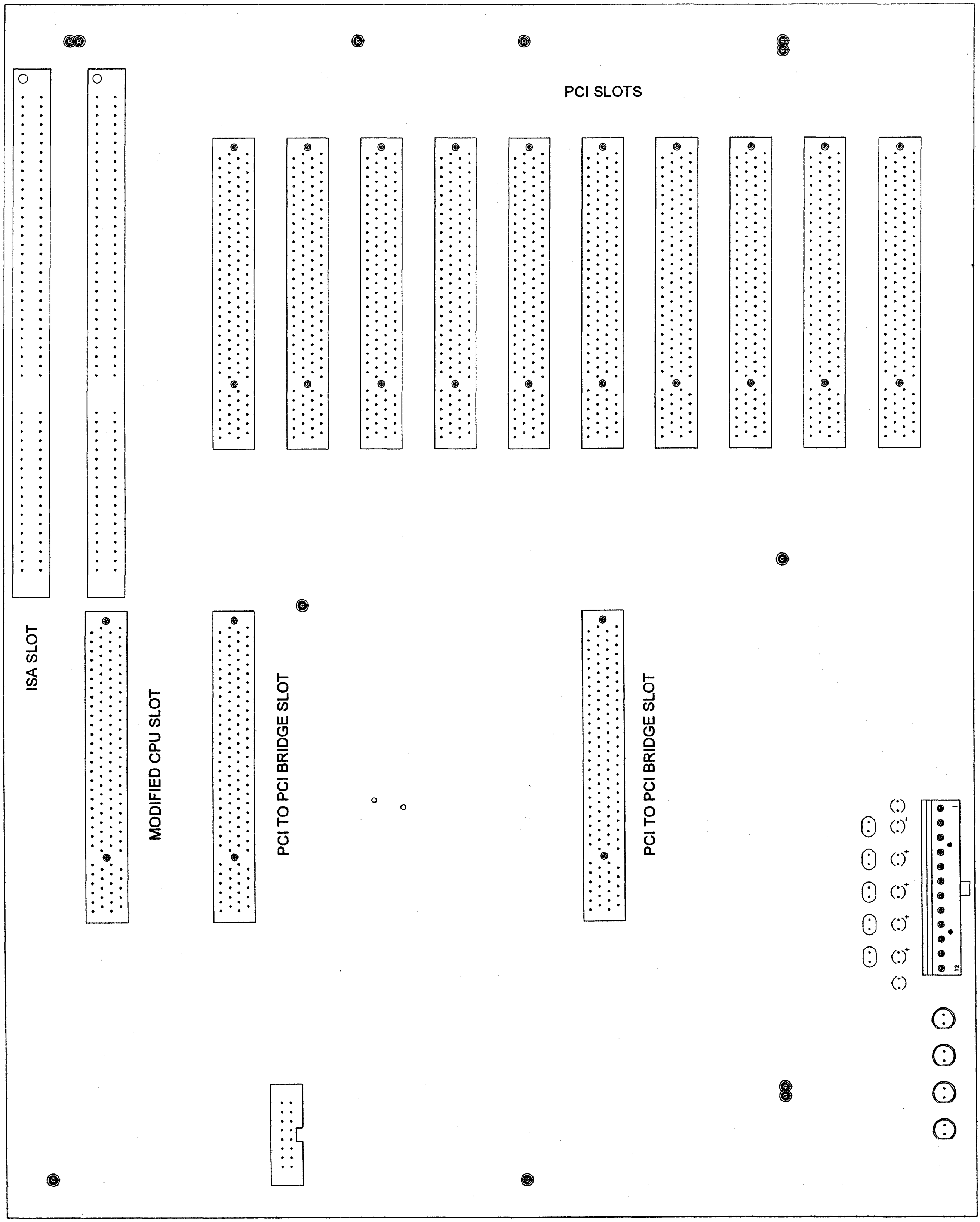

ISA SLOTS
PCI SLOTS
I/O INTERFACE CONNECTOR
1
12
X
+5V
+12V
-12V
GND
GND
GND
GND
-5V
+5V
+5V
+5V
MODIFIED CPU SLOT
HD LED
ON LED
KB INHIBIT*
RESET*
-
SPKR
-
KBCLK
KBDAT
X
KBGND
KBPWR
DOWNLD*
DLGND

Figure 2


PCI SLOTS
ISA SLOT
MODIFIED CPU SLOT
PCI TO PCI BRIDGE SLOT
PCI TO PCI BRIDGE SLOT
1
12

Alpha RISC PC Example Design
PC Motherboard for the 21064A

Tim Miller
Product Line Mgr., Alpha CPU Products
Digital Semiconductor, DEC
77 Reed Rd., MS HL02-2/N7
Hudson, MA 01749-2895
508-568-4122

Abstract

This paper describes the Alpha System Architecture for the 21064A and cache subsystem. It presents the example design timeline, AlphaPC64 features, the 21072 core logic chipset, chipset features, features of the RISC PC motherboard, , and the power supply. It also describes the alpha cache SIMMs, AlphaPC64 flash memory support, and AlphaPC64 firmware support.

Digital Semiconductor

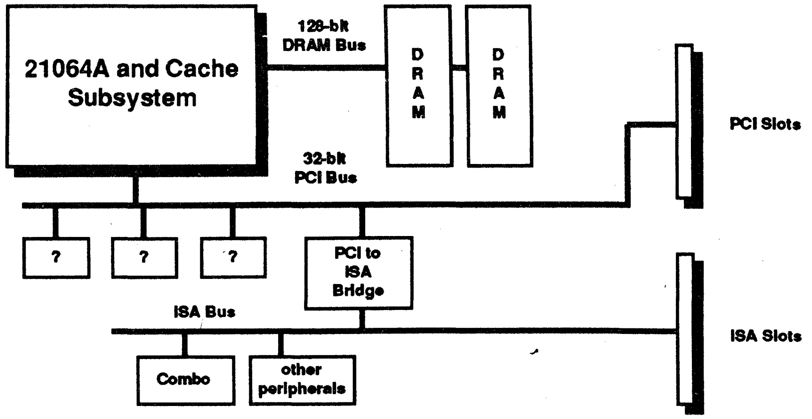
Alpha RISC PC Example Design

PC Motherboard for the 21064A

digital

Digital Semiconductor

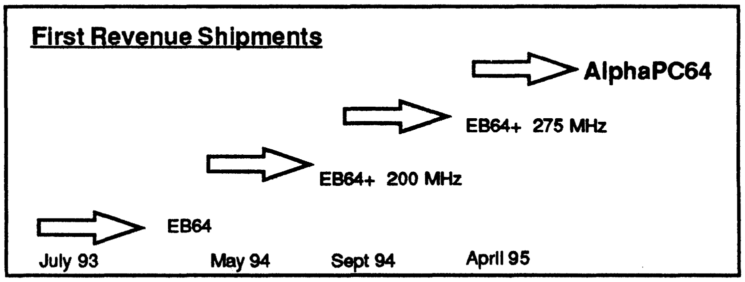
Alpha System Architecture for the 21064A



Digital Semiconductor

21064A Example Design Timeline

Board	Alpha	MHz	
EB64	21064	150	First Design, PLD Based
EB64+	21064	200	Using 21072 Core-Logic Chipset
EB64+	21064A	275	Speed Upgrade
AlphaPC64	**21064A**	**275**	**Cost Reduced, New Design**


First Revenue Shipments

AlphaPC64

EB64+ 275 MHz

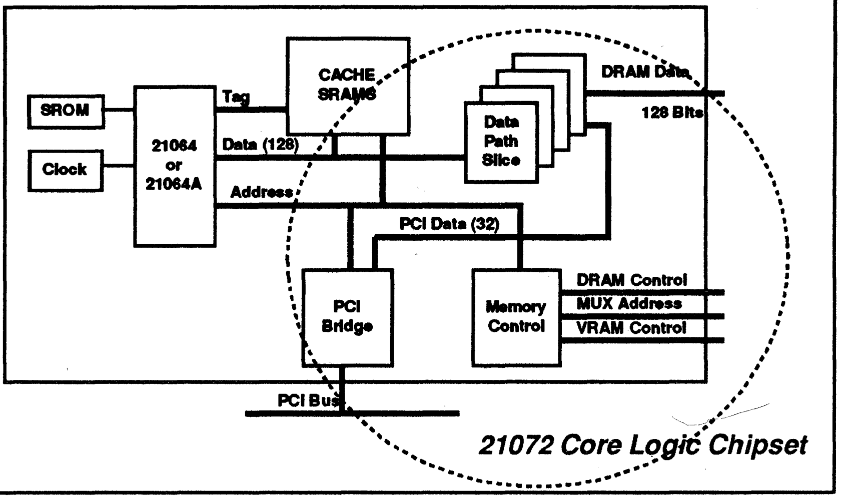
EB64+ 200 MHz

EB64

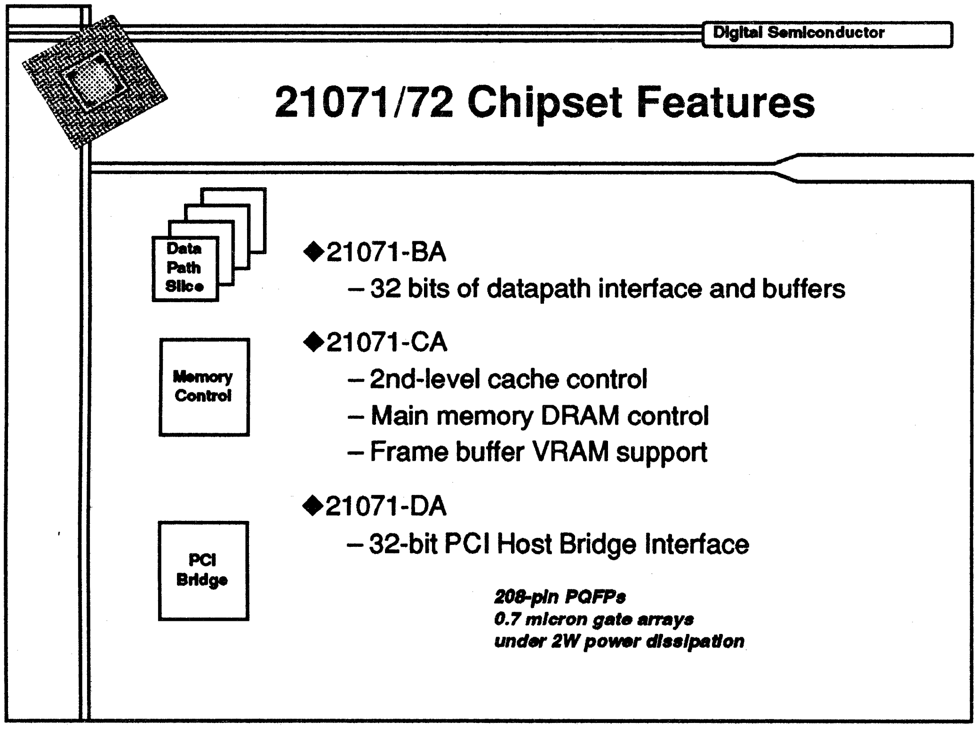
July 93 May 94 Sept 94 April 95

Digital Semiconductor

EB64+
21064A and Cache Subsystem



21072 Core Logic Chipset

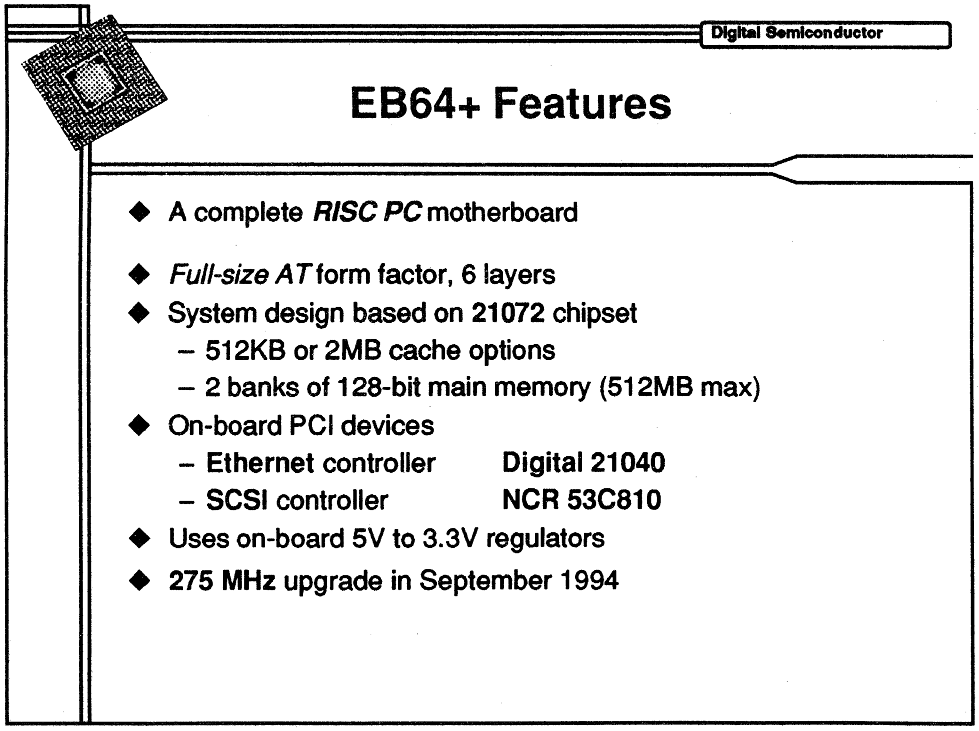

Data
Path
Slice
Memory
Control
PCI
Bridge

Digital Semiconductor

21071/72 Chipset Features

- ◆21071-BA
  - –32 bits of datapath interface and buffers
- ◆21071-CA
  - –2nd-level cache control
  - –Main memory DRAM control
  - –Frame buffer VRAM support
- ◆21071-DA
  - –32-bit PCI Host Bridge Interface

208-pin PQFPs
0.7 micron gate arrays
under 2W power dissipation



Digital Semiconductor

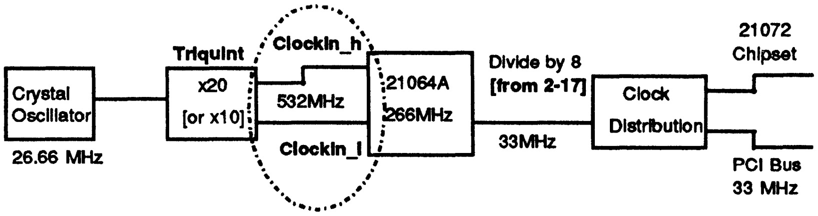
EB64+ Features

- ◆ A complete *RISC PC* motherboard
- ◆ *Full-size AT* form factor, 6 layers
- ◆ System design based on 21072 chipset
  - – 512KB or 2MB cache options
  - – 2 banks of 128-bit main memory (512MB max)
- ◆ On-board PCI devices
  - – Ethernet controller Digital 21040
  - – SCSI controller NCR 53C810
- ◆ Uses on-board 5V to 3.3V regulators
- ◆ 275 MHz upgrade in September 1994

Digital Semiconductor

EB64+ Design Accomplishments

- ◆ PCI has become the industry-standard IO Bus
- ◆ ISA provides for *legacy* devices
- ◆ Low-cost thermal management solution
  - – Heatsink, or Heatsink and Fan
- ◆ Development of high-frequency PLL for Alpha
  - – **Triquint** chip reduces cost

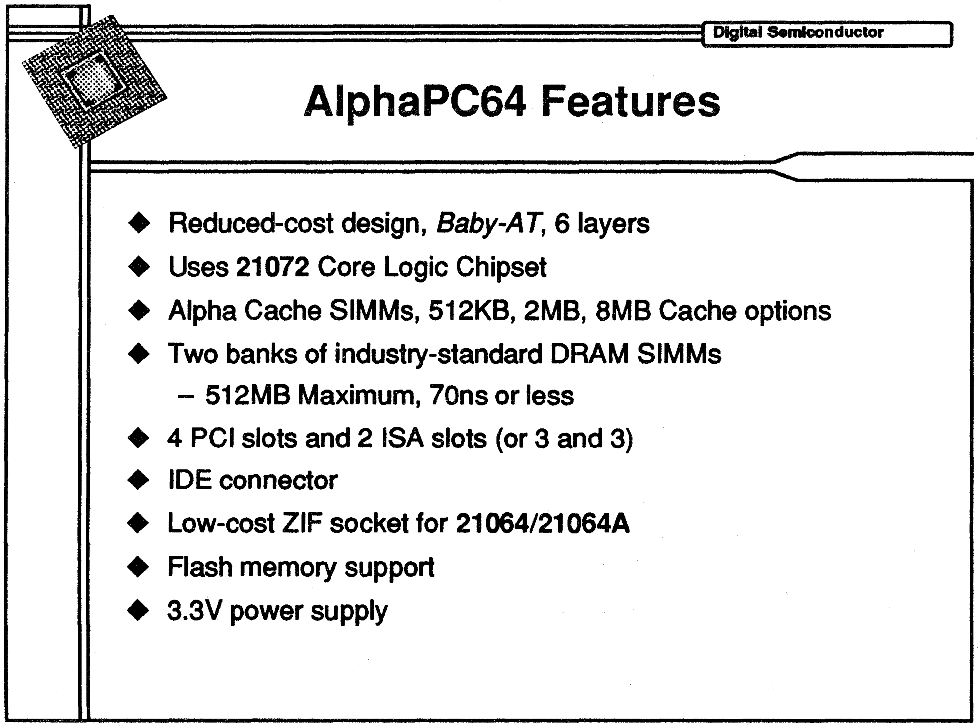


Very short isolated traces

Digital Semiconductor

AlphaPC64 Design Motivation

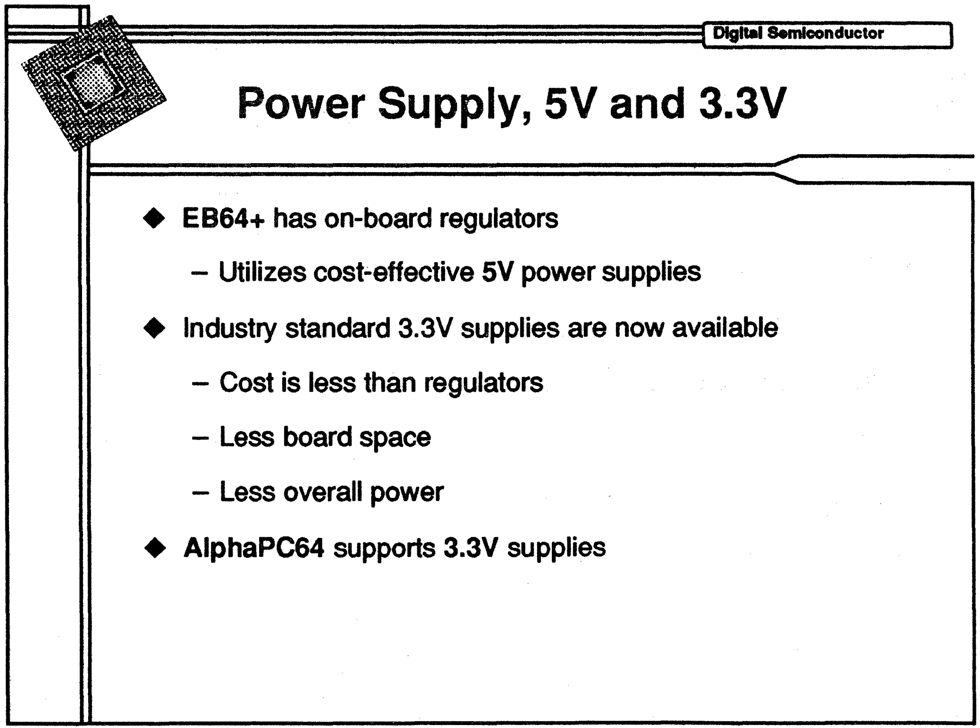
- ◆ Customers require a cost-effective reference design
- ◆ Can we fit into **Baby-AT**, 6 layers?
- ◆ Need to meet FCC requirements
- ◆ Would like to vary cache size and speed
- ◆ Not everybody needs on-board SCSI and Ethernet
  - – More PCI slots



Digital Semiconductor

AlphaPC64 Features

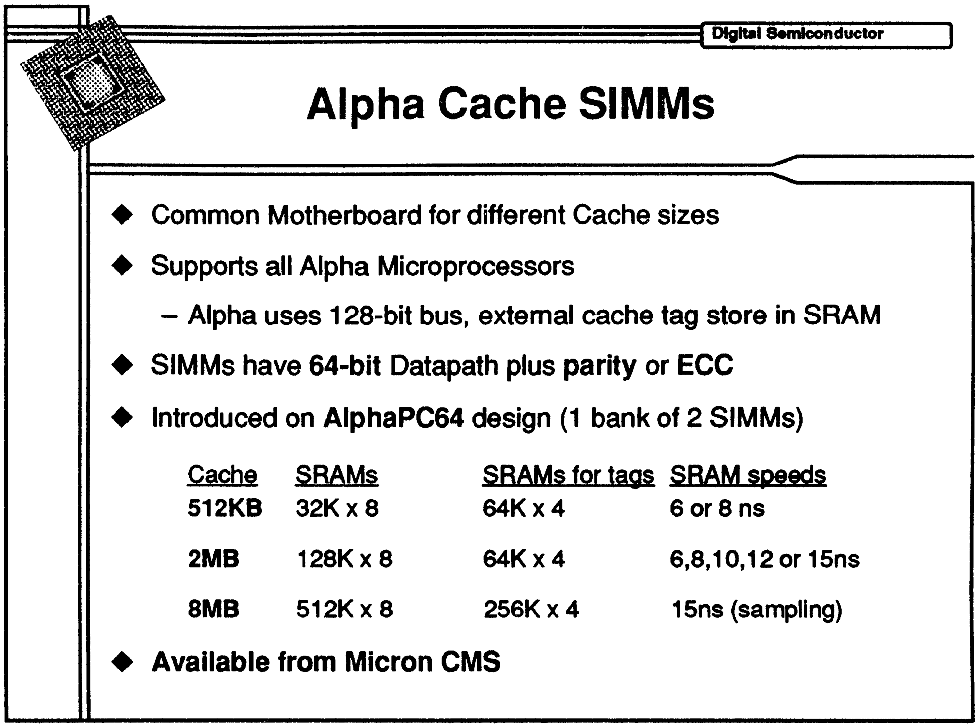
- Reduced-cost design, *Baby-AT*, 6 layers
- Uses 21072 Core Logic Chipset
- Alpha Cache SIMMs, 512KB, 2MB, 8MB Cache options
- Two banks of industry-standard DRAM SIMMs
  - 512MB Maximum, 70ns or less
- 4 PCI slots and 2 ISA slots (or 3 and 3)
- IDE connector
- Low-cost ZIF socket for 21064/21064A
- Flash memory support
- 3.3V power supply



Digital Semiconductor

Power Supply, 5V and 3.3V

- EB64+ has on-board regulators
  - Utilizes cost-effective 5V power supplies
- Industry standard 3.3V supplies are now available
  - Cost is less than regulators
  - Less board space
  - Less overall power
- AlphaPC64 supports 3.3V supplies



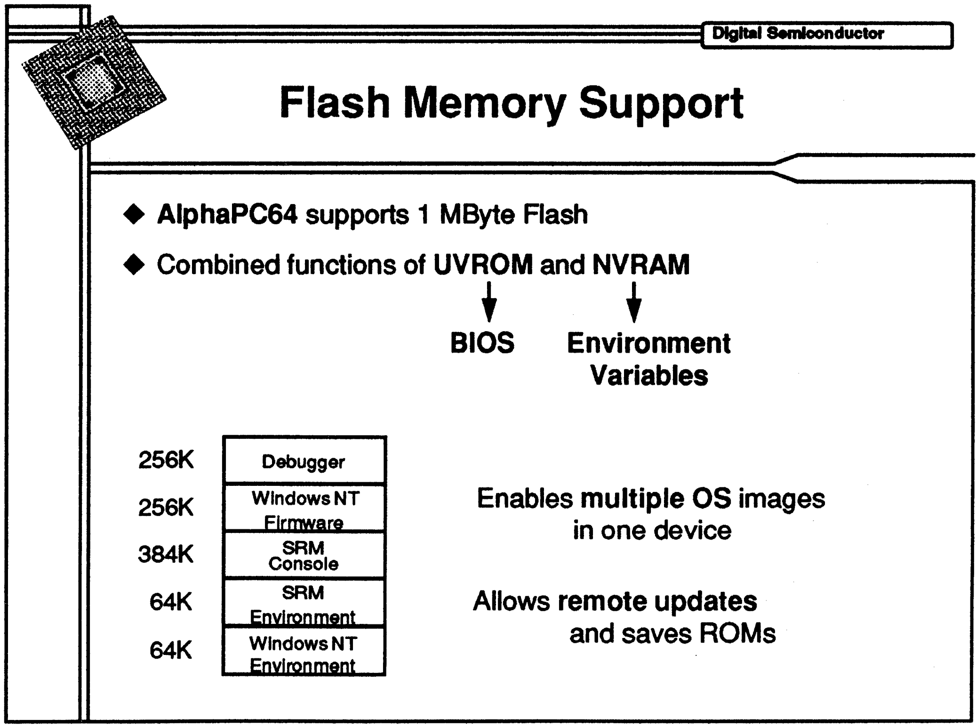
Digital Semiconductor

Alpha Cache SIMMs

- ◆ Common Motherboard for different Cache sizes
- ◆ Supports all Alpha Microprocessors
  - – Alpha uses 128-bit bus, external cache tag store in SRAM
- ◆ SIMMs have **64-bit** Datapath plus **parity** or **ECC**
- ◆ Introduced on **AlphaPC64** design (1 bank of 2 SIMMs)

Cache	SRAMs	SRAMs for tags	SRAM speeds
512KB	32K x 8	64K x 4	6 or 8 ns
2MB	128K x 8	64K x 4	6,8,10,12 or 15ns
8MB	512K x 8	256K x 4	15ns (sampling)

- ◆ **Available from Micron CMS**



Digital Semiconductor

Flash Memory Support

- ◆ **AlphaPC64** supports 1 MByte Flash
- ◆ Combined functions of **UVROM** and **NVRAM**

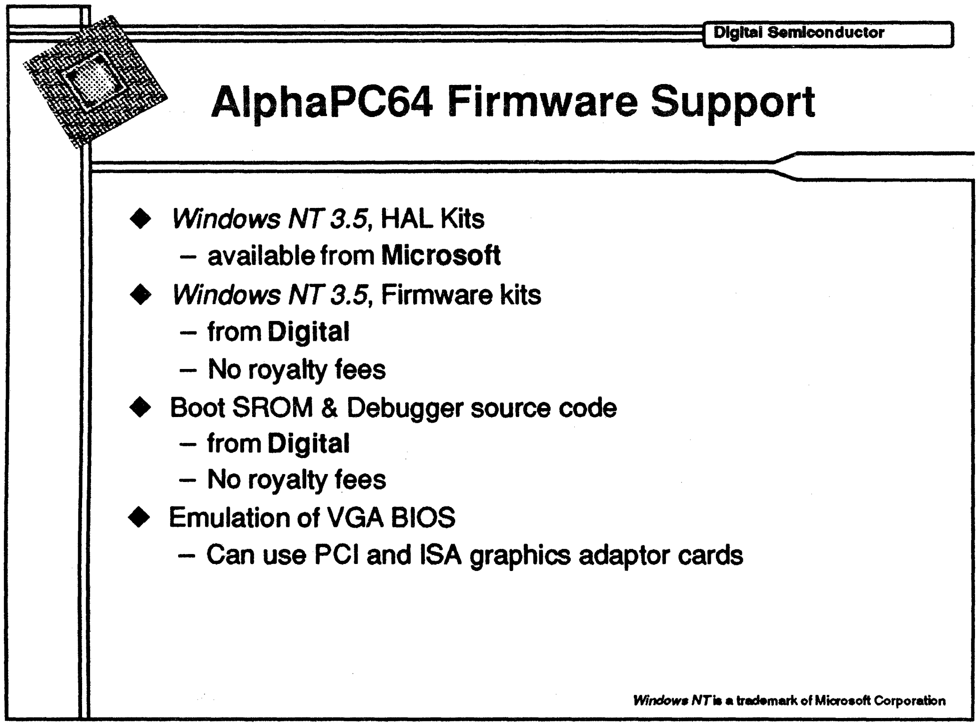
UVROM ↓ **BIOS**

NVRAM ↓ **Environment Variables**

256K	Debugger
256K	Windows NT Firmware
384K	SRM Console
64K	SRM Environment
64K	Windows NT Environment

Enables **multiple OS** images in one device

Allows **remote updates** and saves ROMs

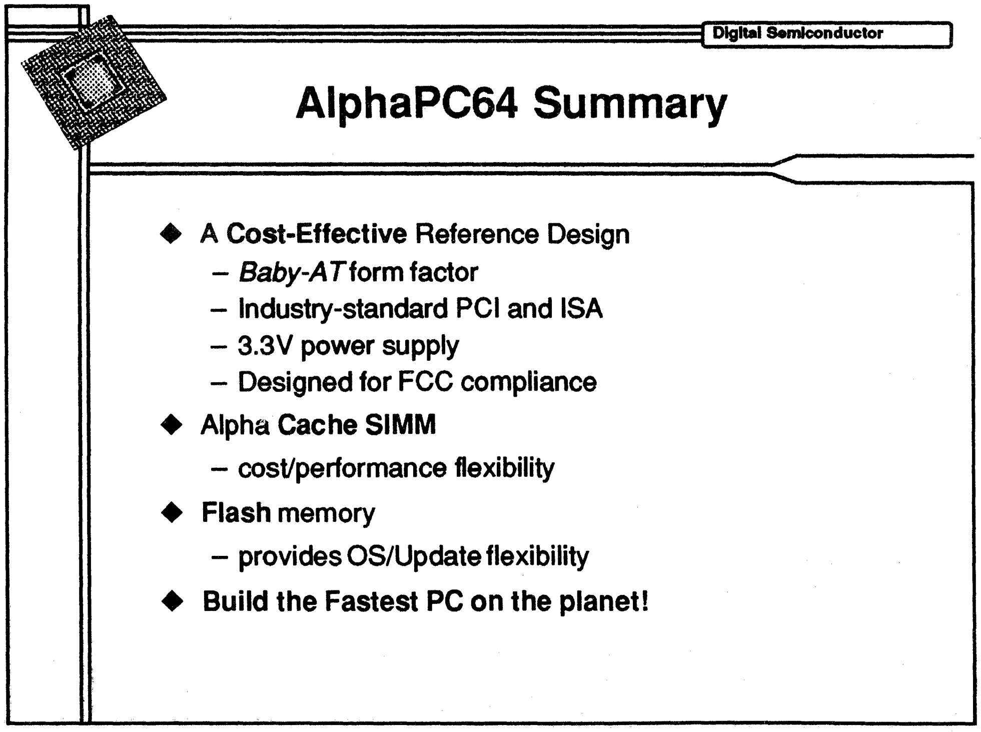


Digital Semiconductor

AlphaPC64 Firmware Support

- ◆ *Windows NT 3.5*, HAL Kits
  - – available from **Microsoft**
- ◆ *Windows NT 3.5*, Firmware kits
  - – from **Digital**
  - – No royalty fees
- ◆ Boot SROM & Debugger source code
  - – from **Digital**
  - – No royalty fees
- ◆ Emulation of VGA BIOS
  - – Can use PCI and ISA graphics adaptor cards

Windows NT is a trademark of Microsoft Corporation



Digital Semiconductor

AlphaPC64 Summary

- ◆ A **Cost-Effective** Reference Design
  - – *Baby-AT* form factor
  - – Industry-standard PCI and ISA
  - – 3.3V power supply
  - – Designed for FCC compliance
- ◆ Alpha **Cache SIMM**
  - – cost/performance flexibility
- ◆ **Flash** memory
  - – provides OS/Update flexibility
- ◆ **Build the Fastest PC on the planet!**

PCI Enhances the Flexibility of VMEbus-Based Single Board Computers

R. Baxter, J. Gipper, G. Novak, C. Pham, M. Rush
Motorola Computer Group
2900 South Diablo Way
Tempe, AZ 85282, USA

ABSTRACT

This paper will discuss the design techniques used to implement a complete PCI I/O subsystem on a PowerPC based single board computer. The paper will provide an overview of the PCI specification itself along with complementary information concerning the IEEE P1386.1 specification for PCI Mezzanine Cards (PMC). The technical advantages of flexible designs using both PCI compatible intelligent controller components and PMC's will also be discussed.

INTRODUCTION

The Peripheral Component Interconnect (PCI) local bus has become a common design element in virtually all high volume, low cost personal computer and single board computer designs. The PCI bus is used primarily as a component interconnect for the single board computer designs but its versatility as an expansion bus has also made it a popular choice with today's designers.

The PCI local bus was specified to establish a high performance local bus standard for many generations of product. The PCI specification provides a selection of features that can achieve multiple price-performance points and can enable functions that allow differentiation at the system and component level.

PCI ON VME

PCI provides many features and benefits for single board computer designs.

Table 1: PCI Features and Benefits

Table 1: PCI Features and Benefits

Feature	Benefit
High Performance	• Transparent upgrade from 32-bit data path (132MB/s peak) to 64-bit data path (264MB/s peak). • Variable length linear and toggle mode bursting for both read and write. • Low latency random accesses. • Capable of full concurrency with processor/memory subsystems. • Synchronous bus with operation from 25 to 33 MHz. • Hidden (overlapped) central arbitration.
Low Cost	• Optimized for direct silicon (component) interconnections using standard ASIC technologies. • Multiplexed architecture reduces pin count and package size of PCI components making economical use of a relatively small number of signals. • Ability to use low cost commodity components due to the wide acceptance of PCI technology.
Longevity	• Processor independent. • Supports both 64-bit address and data extensions. • Both 5V and 3.3V signalling environments are specified.
Ease of Use	• Enables full auto configuration support of PCI local bus add-in boards and components. PCI provides control registers with the device information required for configuration. (Plug and Play)
Interoperability & Reliability	• Forward and backward compatibility of 32-bit and 64-bit add-in boards and components.
Flexibility	• Full multi-master capability allowing any PCI initiator peer-to-peer access to any PCI initiator/target. • Supports up to 256 devices as well as a hierarchy of PCI buses and expansion bus capability.
Data Integrity	• Provides parity on data address, and control lines allowing implementation of robust client platforms.
Software Compatibility	• PCI components can be fully compatible with existing driver and applications software. Device drivers can be portable across various classes of platforms.

The ideal standard must perform a delicate balancing act. It must be stable, avoiding constant revision updates that prevent a vendor from recouping the cost of each design. But it must also change, growing to meet dynamic market needs. Improper trade-offs on this challenging standards hi wire has caused many a proposal to plummet into oblivion.

The PCI standard supports powerful yet cost effective implementations. The signals required to interface to PCI have been kept to a minimum-- under 50 for either a basic target *or* initiator implementation. This reduces component costs, PC board real estate devoted to chips as well as bus traces, not to mention design time.

Performance was certainly a key factor in developing the standard. Options like 64-bit address and/or data bus and multi-master configurations will become checklist items in a few years. Features like the required support of burst transfers by all devices and hidden bus arbitration encourage the design of high performance systems.

Yet the interface is truly processor independent. Various X86 and Alpha AXP microprocessors already have PCI interfaces directly on the processors. Other microprocessor families will certainly provide this interface as its popularity continues to grow. Other chip designers are also adopting this standardized interface with its associated technical and market benefits. Many peripheral I/O controllers already connect directly to PCI. Besides saving glue logic, this gives the user access to the full functionality of each controller chip. The board designer need not waste valuable time re-inventing interfaces to each controller chip that will utilize its unique features. Rather, the chip vendors themselves provide the PCI interface most applicable to their specific controller or CPU.

Both board and system designers benefit from this standardized interface at the component level and at the expansion bus level. On-board local devices as well as add-in cards are supported. This expansion bus capability allows very versatile system solutions. Besides factory installed option support, it allows the user to customize his system as his computing needs change.

But while PCI is useful now, will it be viable in a few years? PCI provides a performance growth path through 64-bit expansion and the other performance features mentioned previously. It also supports 3.3 V operation for lower power designs. PCI addressing supports up to 256 devices, far more than is required today. A hierarchy of PCI buses is expands this support even further. The standard specifies configuration registers to provide software with a standardized way of initializing the system. While not that difficult to support from a hardware viewpoint, these registers help simplify the complex task of auto-configuration support.

PCI MEZZANINE CARDS WITH VME

The VMEbus was developed to provide systems designers with the electrical and mechanical standards to interconnect boards to form a system. VMEbus provided a common ground for board and system designers and resulted in the development of many products by numerous vendors. The competitive environment caused lower costs and a successful VMEbus market. The PCI bus takes this strategy to the board level. Today there are many PCI devices being developed by numerous companies and are used in virtually all personal and single board computer designs. Lower costs have resulted because the bus interface does not need to be reinvented each time a new microprocessor enters the market and glue logic is not required.

A key element that has been missing from VMEbus designs since the inception of VMEbus has been a standard for peripherals and peripheral expansion from the main boards. Several attempts have been made in the past several years to define a standard that could provide this expansion. Many developers of VMEbus products attempted to define and implement their own architectures that they hoped would be used as industry standards. Sbus was perhaps the most developed of those solutions. None of the numerous solutions ever gained industry wide acceptance.

The advent of PCI excited the industry to the point that both the VMEbus and the Multibus communities joined forces to develop a standard that is quickly becoming the industry choice for mezzanine expansion. The PCI Mezzanine Card (PMC) standard IEEE P1386.1 is that standard. PMC combines the electrical elements of the PCI local bus with a set of common mechanical definitions, IEEE P1386, the Common Mezzanine Card (CMC) to form the basis of PMC.

PMC is an excellent solution because it offers the features and benefits of the PCI local bus plus:

Table 2: PMC Features and Benefits

Feature	Benefit
Form Factor	• A well defined mechanical form factor for use on may different base boards. • Low profile, fits in a single VME slot. • Single and double wide modules allow flexibility in the amount of board space used.
I/O Options	• Allows connection to be made to the front panel of a VME module or to be routed to P2 for VME host boards supporting this feature.
Electrical Interface	• A well defined electrical interface which is also a device level interface. A PCI component placed on a PMC card does not require another bridge device to interface the chip to the bus.
32- or 64-bit PCI Local Bus	• Plenty of bus bandwidth for high performance and demanding applications.
Mechanical Keying	• Prevents association of host and mezzanine card with incompatible signaling voltages.

IEEE P1386, the Common Mezzanine Card (CMC) defines the mechanical standards for the entire IEEE P1386 family. This standard includes details like the 4 board sizes allowed, component height restrictions, the physical connector, voltage keying, and power consumption. The PCI Mezzanine Card (PMC) and Sbus Mezzanine Card (SMC) are the 2 currently defined families that comply with this specification. PMC is defined by the IEEE P1386.1 specification. Since all of the mechanical definition is provided by the related CMC specification, IEEE P1386.1 only defines the specific PCI electrical signals associated with the pins on each connector.

Mezzanine buses added another level of complexity to board design. Many mezzanine buses have been defined over the years and most have failed to gain wide acceptance. One reason is the mezzanine bus was not the same as the board's local bus and required a bridge device to connect the buses. Plus, mezzanine buses were not supported at the device level which meant that additional glue logic was required to connect the mezzanine bus to the devices on the mezzanine board. Sbus was well accepted and used as a local bus but peripheral chips with an Sbus interface were not widely developed and bridge chips were often required. In general, bridges require expensive real estate, impair performance and impact software. The same device, such as a SCSI or Ethernet controller used on the main board may appear very different to the software when used on a mezzanine board. Because the PMC uses the PCI electrical specification, bridge devices are not required and the devices will appear the same to the software drivers. This reduces the software development time, improves performance and reduces time to market.

PMCs can easily be implemented on other form factors, such as standard PCI cards, because the electrical PCI interface does not change. PMCs can also be adapted to many other form factor hosts besides VME. A standardized mezzanine card allows many different vendors to provide cards for many different systems. This plays a role in developing larger economies of scale that drive down costs.

DESIGN ISSUES WITH PCI

Single board computer designs, whether they are VMEbus or motherboard designs, have certain criteria that must be met if the design is to meet the market requirements and be a success. The major criteria and the goals are listed in the following table.

Table 3: Design Criteria

Criteria	Goals
Low Cost	Use industry standard components to drive high economies of scale that maintain a competitive price advantage
High Performance	A local peripheral bus that would support today's data transfer requirements. Single board computer designs need to support multiple peripherals on the board as well as expansion. Bus loading is a concern.
Future Growth Path	A solution that is at its maximum level at implementation is not acceptable in terms of future requirements and capabilities.
Industry Acceptance	Industry acceptance is required to drive price and selection of components.
Large Selection of Components	A large selection gives designers a choice. These choices allow the product to be optimized for the target applications and markets.
Software Support	Software support is critical to reducing design cycle times. With today's complex peripheral controllers, the software development burden can add substantial delay to a project.
System Expansion	It is desirable for the architecture to support off-board peripheral expansion.

A standardized interface is important to board designers. In the past, they were forced to interface chips with different bus interfaces to the local bus. Incompatible interfaces resulted in many "bus bridges" being developed. Many of the bus incompatibilities were difficult to overcome and resulted in lower performance. The PCI bus architecture requires a single bridge between the MPU bus and the PCI bus. Since real estate is limited on VME boards and other standardized form factors, many chips were chosen for their bus interface rather than for their programming interface, features or performance.

The custom interfaces required for each device may be different each time the device is used and this resulted in different software drivers being developed each time the device was instantiated. With PCI, the hardware interface is standard and the chip should always look the same to the software driver. Therefore, a standardized hardware interface also leads to a standardized software interface.

The PCI specification defines a 64-bit extension which will provide performance improvements and ensure a long life for the PCI bus. 3.3 volt operation is also addressed. This is also important for a long life since more systems and devices will move to 3.3 volts in the future.

Big endian byte ordering versus little endian byte ordering is a major issue when designing with PCI. The PCI bus is little endian so when the PCI bus is used with a big endian processor and programs, the software must provide the translation. This is amplified even more in a VMEbus application because VMEbus is big endian. With a VMEbus design it is quite possible to go from an initiator VMEbus board with a big endian processor to little endian PCI to big endian VMEbus and back up the chain on the target end of the transaction. Extreme caution is needed to be sure that the endian impact is fully understood. Software incurs a much larger burden than might normally be expected.

The PCI bus is not a backplane bus. The PCI bus was defined to be driven by low power CMOS devices, therefore there are severe loading restrictions when compared to a backplane bus such as the VMEbus. The PCI bus may be extended using repeaters, but this will impact performance. The PCI bus is a very good local bus and provides limited expansion but at the present time it is not a backplane bus replacement. This is a major issue when users of the board products start to realize the performance capability of PCI. They see an opportunity to expand the PCI to external peripherals via PCI bridges. What is not understood is the impact of multiple bridges on the PCI bus.

SOFTWARE CONSIDERATIONS

Software developers welcome the PCI bus architecture as a breath of fresh air. A look at some of the key features of this architecture from a software perspective is important when considering PCI bus.

A common software design goal today is to maintain an "open" systems perspective. With the variety of VMEbus modules on the market today, this open systems perspective is difficult to maintain. VMEbus modules, as a whole, are difficult to identify programmatically. However, it is required that supported VMEbus modules be configured through a list of known addresses per type and function. Over the life of a software product it is often necessary to add support for new VMEbus modules. During the entire life cycle, this address list becomes difficult to manage and maintain.

The PCI bus architecture allows software to dynamically identify the PCI modules and/or devices within a system. This identification process is the same for all PCI compliant devices. The probing of PCI devices doesn't require the software probing functions to be capable of exception/interrupt (e.g., machine check, bus error) processing. Simple MPU read cycles, from the perspective of the software, can be performed through the configuration register space, to determine the presence of a PCI device. The architecture guarantees that the PCI bridge device will return a predefined value indicating no device is present at a particular PCI "slot". If a device is present at the probed PCI "slot", the probed device will reply with data containing its vendor identifier (ID).

The software interface is further standardized by the Configuration Registers. These registers allow software to determine the devices and their revisions in a standardized way.

Each PCI compliant device adheres to predefined configuration register space. This configuration register space (256 bytes total) provides features necessary for system configuration needs. The configuration register space consists of mandatory and optional features. Mandatory features are; device ID, vendor ID, command/status, class code, revision ID, and header type registers.

The class code register is divided into three registers: base class, sub-class, and programming interface. The base class portion of the class code register identifies the basic function of the device (e.g., mass storage, network, display). The "sub-class" portion of the register further identifies the device's function (e.g., mass storage - SCSI, IDE, floppy disk, IPI).

Some of the predefined optional features include: cache line size, latency timer, built-in self-test (BIST), base address, expansion ROM address, interrupt pin/line, and timing information.

The base address register permits the configuration mechanism software to determine the memory and I/O address space requirements, and to programmatically locate the address space, of a PCI device. These address spaces are typically where the device's control (e.g., command/status, data ports) registers are located. A device may have both memory and I/O address spaces. The base address register permits the configuration software to dynamically locate the device's memory and or I/O address spaces. The register also identifies the address space (i.e. memory or I/O) and the size of the address space. This programmability of the location and size of a device's control space eases the burden of the memory management functions, as well as coalescing and localizing all control spaces.

The BIST feature allows initialization and diagnostic software to dynamically determine the health of a PCI device.

The expansion ROM feature allows PCI devices to incorporate their own device ROM. This ROM may contain the BIST, POST, BIOS, device initialization functions, a system boot function, and/or interrupt service routines. The expansion ROM address register behaves similar to the base address register, as discussed previously. A predefined ROM header is specified to aid configuration software in the contents and requirements of the expansion ROM.

The interrupt pin and line feature permits the configuration software to initialize the interrupt routing information. The device drivers can use this information later to determine interrupt priority and vector configuration.

The PCI bus architecture minimizes the burden of configuration software designers and maintainers, and allows a commonality approach without sacrificing a device's uniqueness.

DESIGN ISSUES WITH PMC

The perfect mezzanine card for VMEbus; 1) allows the host to stay within a single VMEbus slot, 2) provides front panel and P2 I/O access, 3) is inexpensive, 4) is widely supported, 5) has high performance capability, and 6) makes it easy to design custom modules. PMC meets these requirements. There are, however, constraints that users and designers of PMCs should consider.

1. **Pn variations/host capabilities.** The PMC mechanical specification allows for four 64-pin connectors on the module. Clearance provisions for P3 and P4 connectors should be considered to allow compatibility with 64-bit hosts and hosts that support the P4 connector for I/O. Not all combinations are necessary but the strategy for supporting them should be well thought out.

2. **Board space/keep out spaces.** There are restrictions on board space and keep out spaces that the host board needs to follow to allow various PMCs to fit on the host. The keep out areas are typically reserved for I/O connectors on the PMC.

3. **Component height limitations.** Tied in with the keep out space restrictions are limitations on component height on both the host and the PMC. It is vital to respect these requirements to allow the boards to mate together. Lack of adequate component clearance prevents a reliable mating of boards. These clearances are fully appreciated during shock and vibration testing

4. **Thermal considerations.** Allowance for proper cooling is a key requirements and closely associated with component height restrictions. Not only is adequate clearance required, but hot spots may require additional airflow considerations during PC board layout.

5. **Voltage signaling.** The PMC specification has provisions for both 5V and 3.3V *signal* levels, as well as the 3.3V and 5V *power* pins. Both interfaces must be defined.

6. **Host specifications.** It is important that developers of host products fully specify the capabilities of their host board. Not all hosts may be able to handle all PMCs. The host specification should clearly state the size, signalling, connector configuration, etc. that it supports so that users can make the right choice in selecting mating PMCs.

Most of these parameters are directly controlled by the host board that carries the PMC. A well designed mezzanine card requires that the carrier be fully understood and characterized.

FUTURE PLANS FOR PCI

For any bus to stay competitive, it must continue to evolve. The VMEbus standard has continued to increase both performance and functionality during its 13 years of growth. Improved block transfer modes, serial bus support, live insertion, 3.3V support and other proposed standards are extending VMEbus to meet the needs of tomorrow's users. An ideal local bus must grow in a similar manner to stay competitive with other local buses and provide performance that compliments VMEbus or whatever the system bus may be. PCI Rev 2.0 is a relatively new standard that has many advanced features which are just starting to be utilized, features such as 64-bit capabilities and open firmware. Nonetheless, PCI is looking to the future with developments like 66 MHz operation.

SUMMARY

PCI brings to the VMEbus technology the ability to design lower cost products with high performance components. This will strengthen the VME bus position in the industrial technical markets. Development cycle times will be reduced for VME products because of the plug-in nature of PCI controller chips. System scalability with PMCs will reduce overall system costs for VME system designs. VMEbus and PCI are in for a long marriage.

REFERENCES

PCI Local Bus Specification, Revision 2.0

Physical and Environmental Layers for PCI Mezzanine Cards, IEEE P1386.1/Draft 1.5

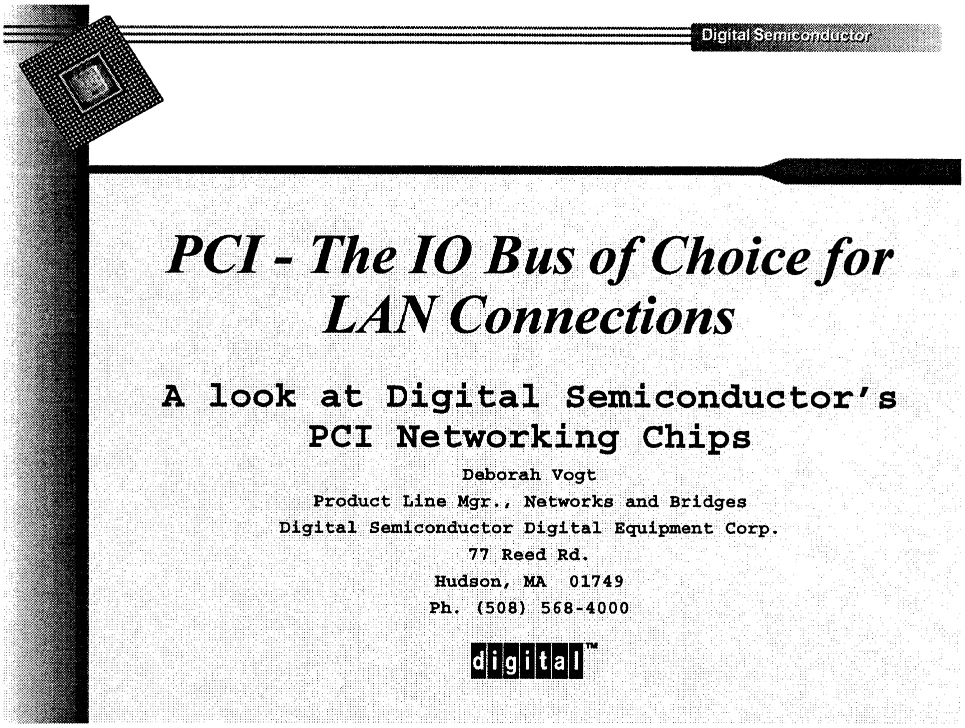
Common Mezzanine Card Family: CMC, IEEE P1386/Draft 1.5

PCI Bus Latency and Network Adapter Design

Glen Gibson
Strategic Marketing Engineer
Advanced Micro Devices
PO Box 3453
Sunnyvale, CA 94088
(408) 732-2400/749-5466 (fax)
glen.gibson@amd.com

Advanced Micro Devices intends to discuss issues of PCI bus latency as it relates to network FIFO and buffer sizes. Latency is a particularly important issue as attention turns from 10 Mbps Ethernet to the newer 100 Mbps versions such as Fast Ethernet. Comparisons will be made to similar situations on the VL-bus.

For more information about AMD's network adapter chips, please contact your local sales office.



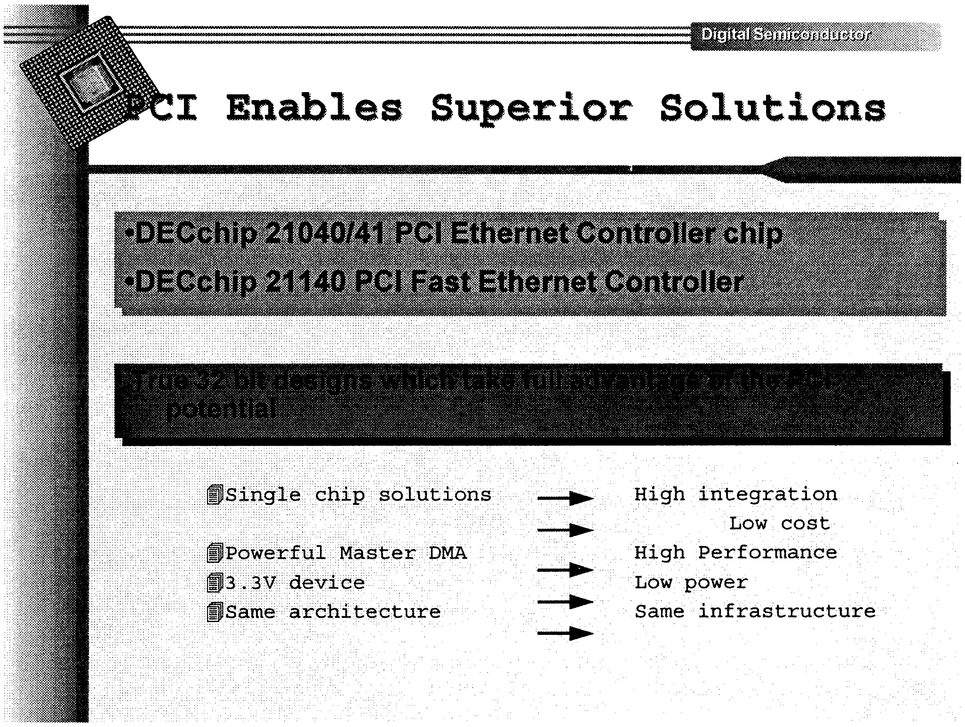
Digital Semiconductor

PCI - The IO Bus of Choice for LAN Connections

A look at Digital Semiconductor's PCI Networking Chips

Deborah Vogt
Product Line Mgr., Networks and Bridges
Digital Semiconductor Digital Equipment Corp.
77 Reed Rd.
Hudson, MA 01749
Ph. (508) 568-4000

digital™



Digital Semiconductor

PCI Enables Superior Solutions

- •DECchip 21040/41 PCI Ethernet Controller chip
- •DECchip 21140 PCI Fast Ethernet Controller

True 32 bit designs which take full advantage of the PCI potential

Single chip solutions → High integration
→ Low cost
Powerful Master DMA → High Performance
3.3V device → Low power
Same architecture → Same infrastructure
→

Digital Semiconductor

Product Specifics

DECchip 21040/21041

- PCI Ethernet Controller
- On-chip transceivers
- Media autodetection
- Full duplex
- JTAG
- 120 pin PQFP

- 21040 shipping in volume since 6/94
- 21041 production in 3/95

DECchip 21140

- PCI Fast Ethernet Controller
- 10/100
- Speed autodetection support
- Full duplex
- MII interface
- JTAG
- 144 pin PQFP

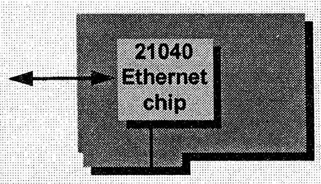
- Production since 12/94

Digital Semiconductor

High Integration => Low cost

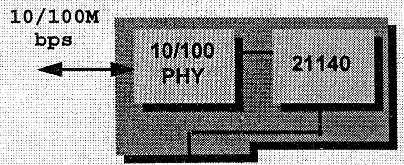
21040/41 (Ethernet)

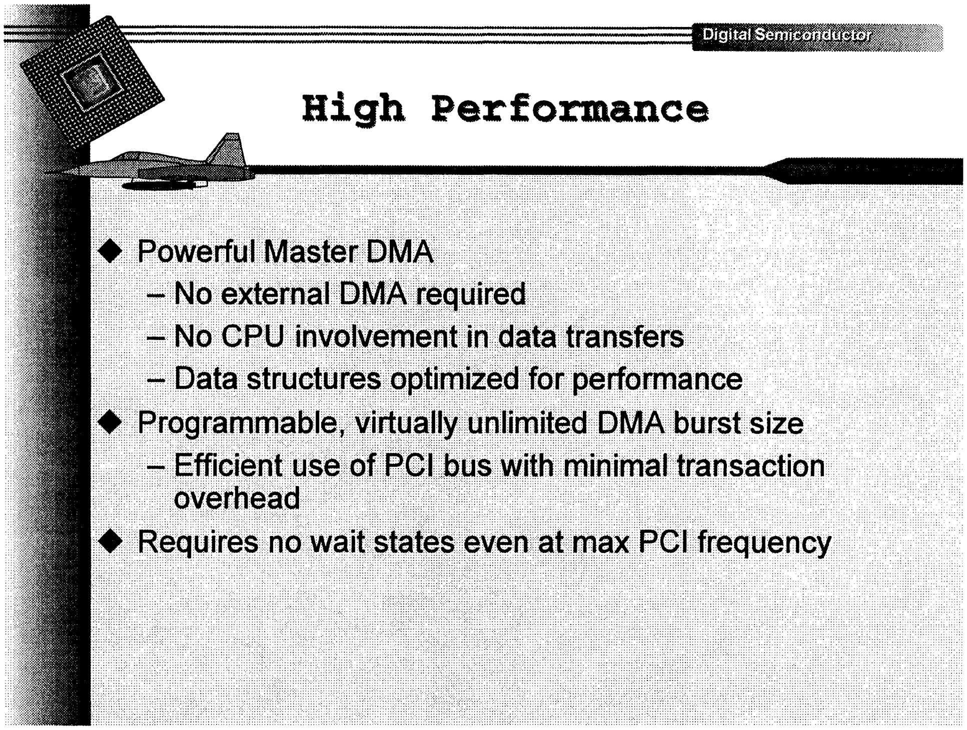
- Single chip solution
  - PCI and system interface
  - All buffering on-chip
  - Integrated AUI, 10BaseT
- 120 pin PQFP



21140 (Fast Ethernet)

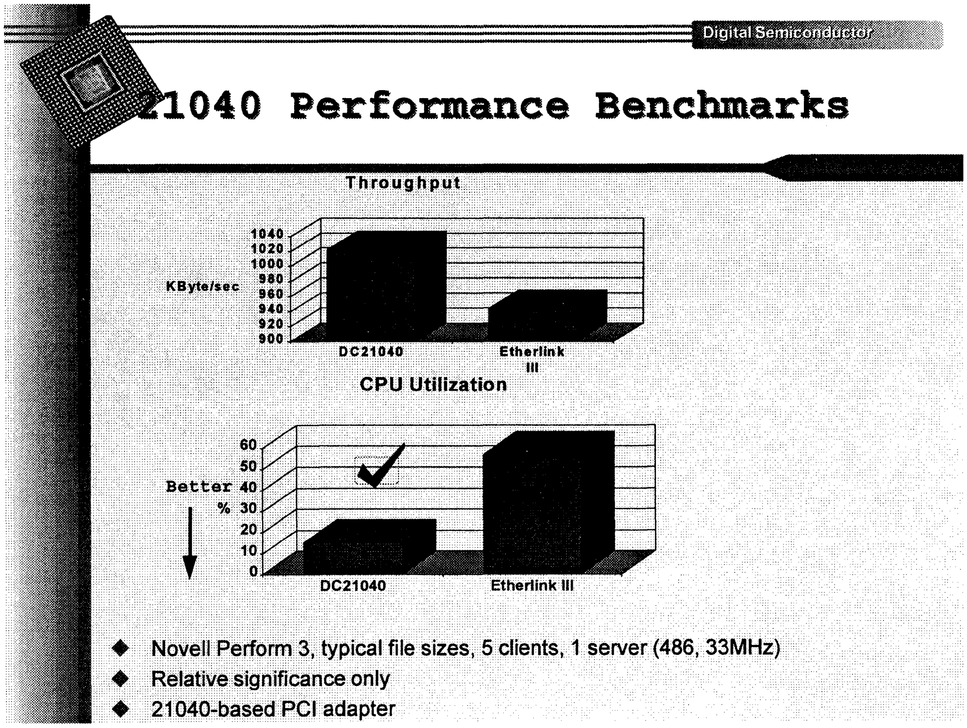
- Only an additional PHY chip required
  - PCI and System Interface
  - All buffering on-chip
  - Integrated scrambler, PCS
- 144 pin PQFP




Digital Semiconductor

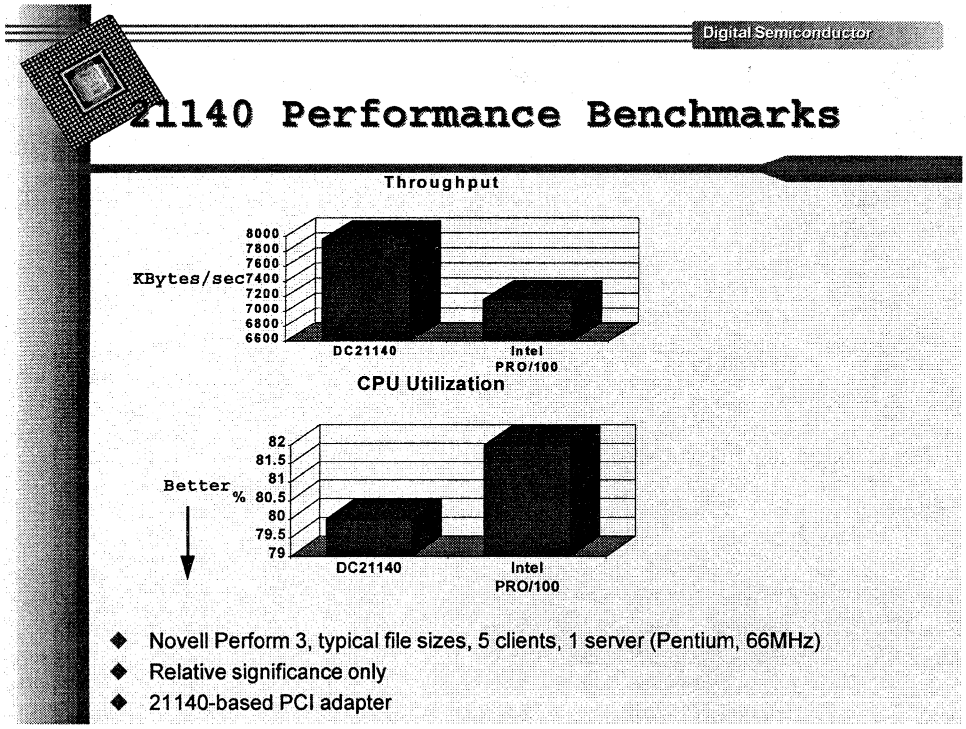
High Performance

- ◆ Powerful Master DMA
  - No external DMA required
  - No CPU involvement in data transfers
  - Data structures optimized for performance
- ◆ Programmable, virtually unlimited DMA burst size
  - Efficient use of PCI bus with minimal transaction overhead
- ◆ Requires no wait states even at max PCI frequency



21040 Performance Benchmarks

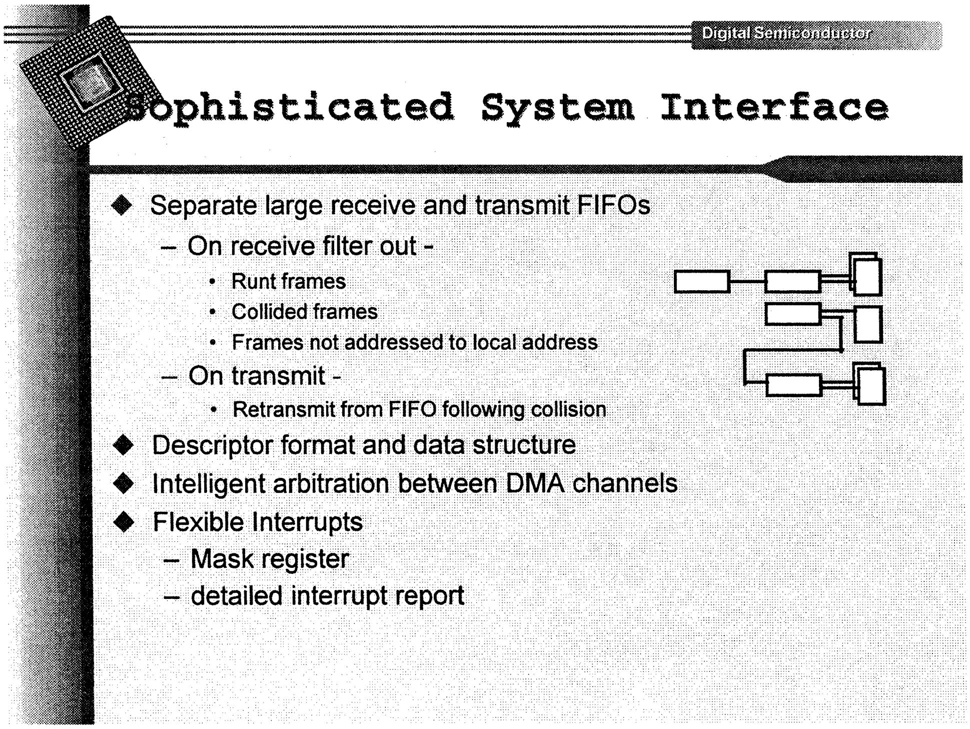
- ◆ Novell Perform 3, typical file sizes, 5 clients, 1 server (486, 33MHz)
- ◆ Relative significance only
- ◆ 21040-based PCI adapter



Digital Semiconductor

21140 Performance Benchmarks

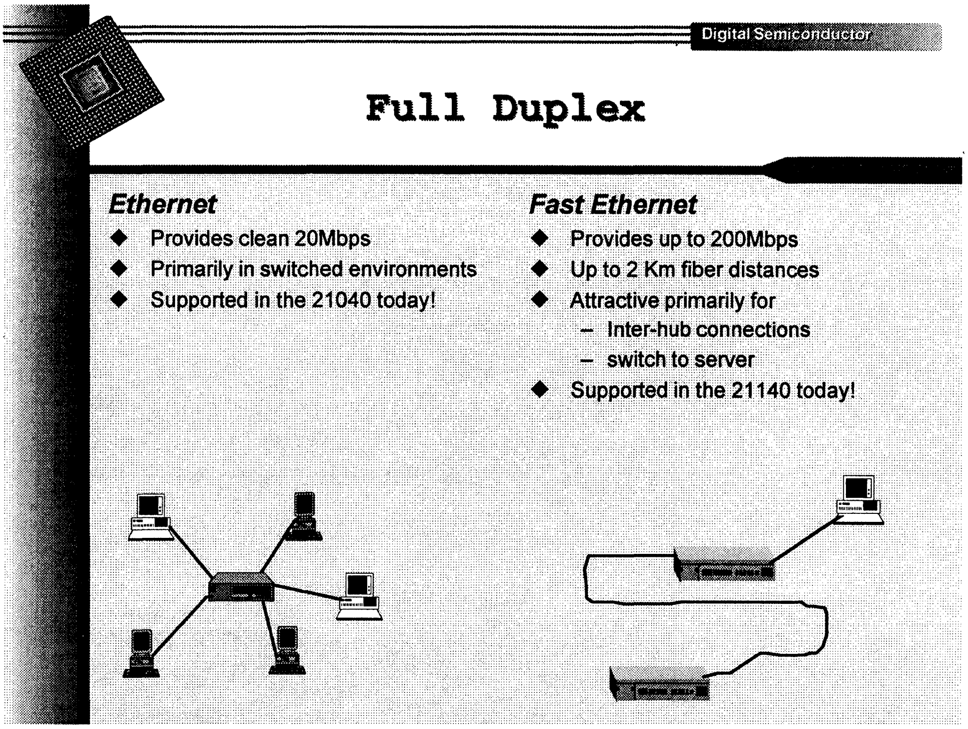
- ◆ Novell Perform 3, typical file sizes, 5 clients, 1 server (Pentium, 66MHz)
- ◆ Relative significance only
- ◆ 21140-based PCI adapter



Digital Semiconductor

Sophisticated System Interface

- ◆ Separate large receive and transmit FIFOs
  - – On receive filter out -
    - Runt frames
    - Collided frames
    - Frames not addressed to local address
  - – On transmit -
    - Retransmit from FIFO following collision
- ◆ Descriptor format and data structure
- ◆ Intelligent arbitration between DMA channels
- ◆ Flexible Interrupts
  - – Mask register
  - – detailed interrupt report



Digital Semiconductor

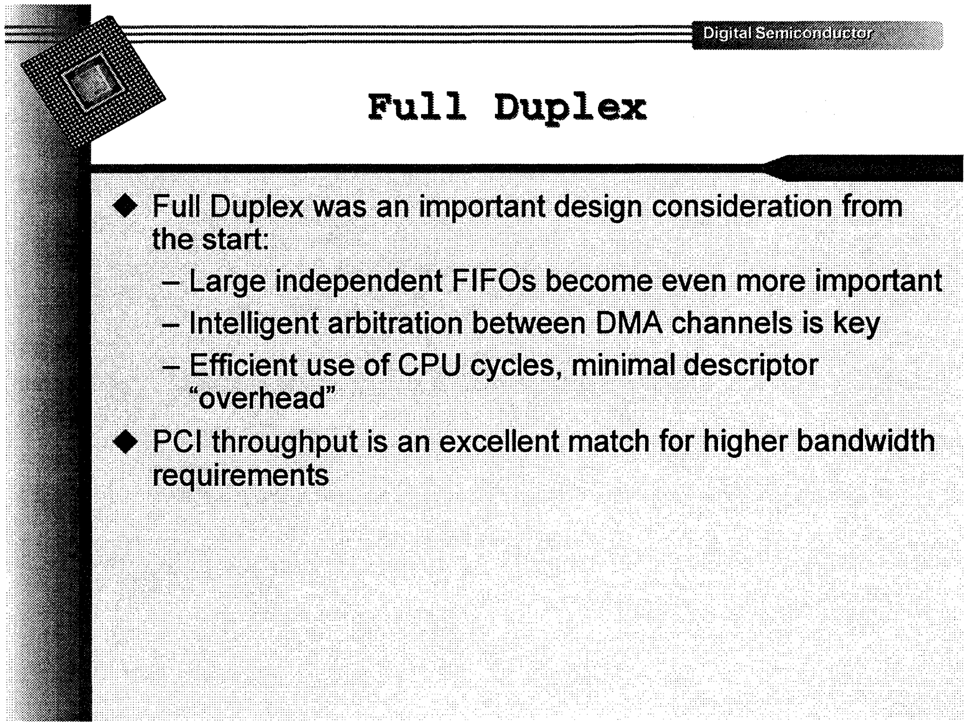
Full Duplex

Ethernet

- Provides clean 20Mbps
- Primarily in switched environments
- Supported in the 21040 today!

Fast Ethernet

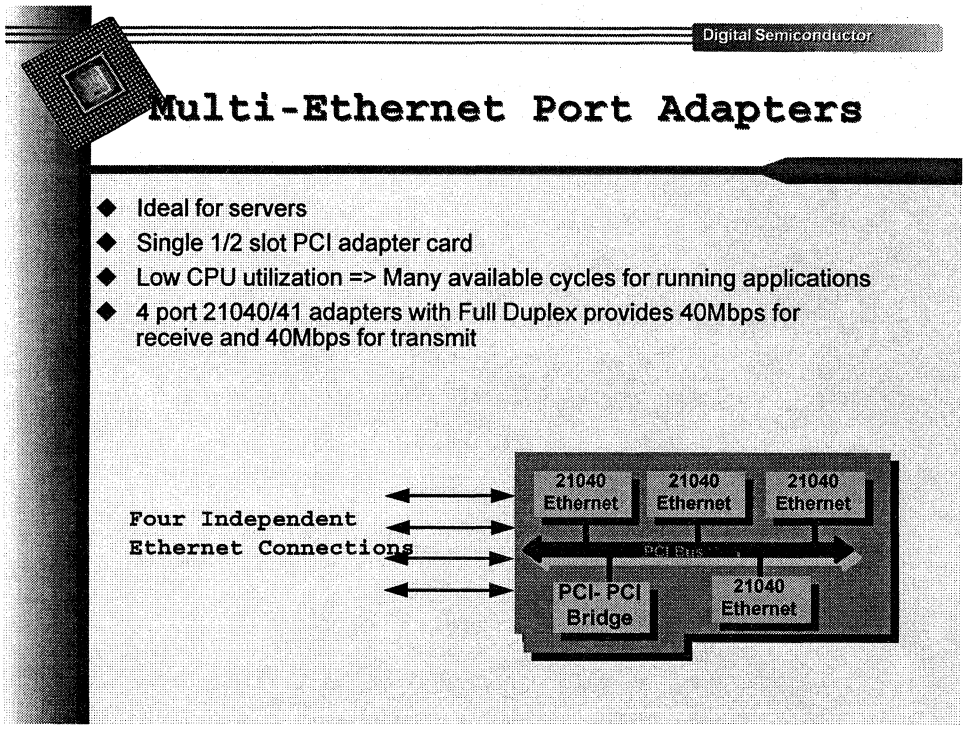
- Provides up to 200Mbps
- Up to 2 Km fiber distances
- Attractive primarily for
  - Inter-hub connections
  - switch to server
- Supported in the 21140 today!



Digital Semiconductor

Full Duplex

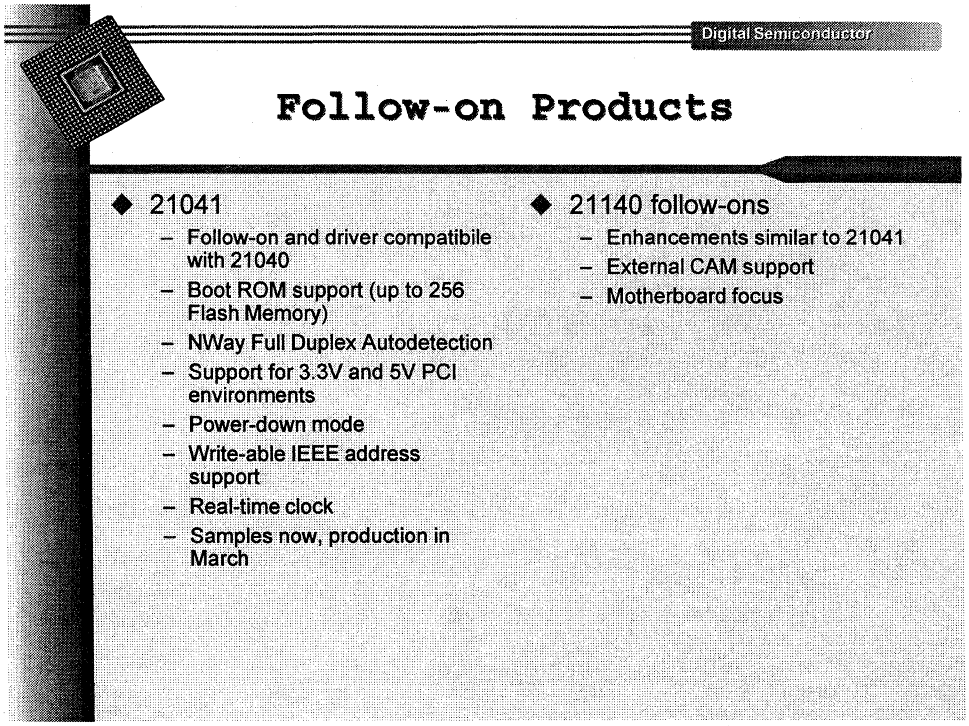
- Full Duplex was an important design consideration from the start:
  - Large independent FIFOs become even more important
  - Intelligent arbitration between DMA channels is key
  - Efficient use of CPU cycles, minimal descriptor "overhead"
- PCI throughput is an excellent match for higher bandwidth requirements



Digital Semiconductor

Multi-Ethernet Port Adapters

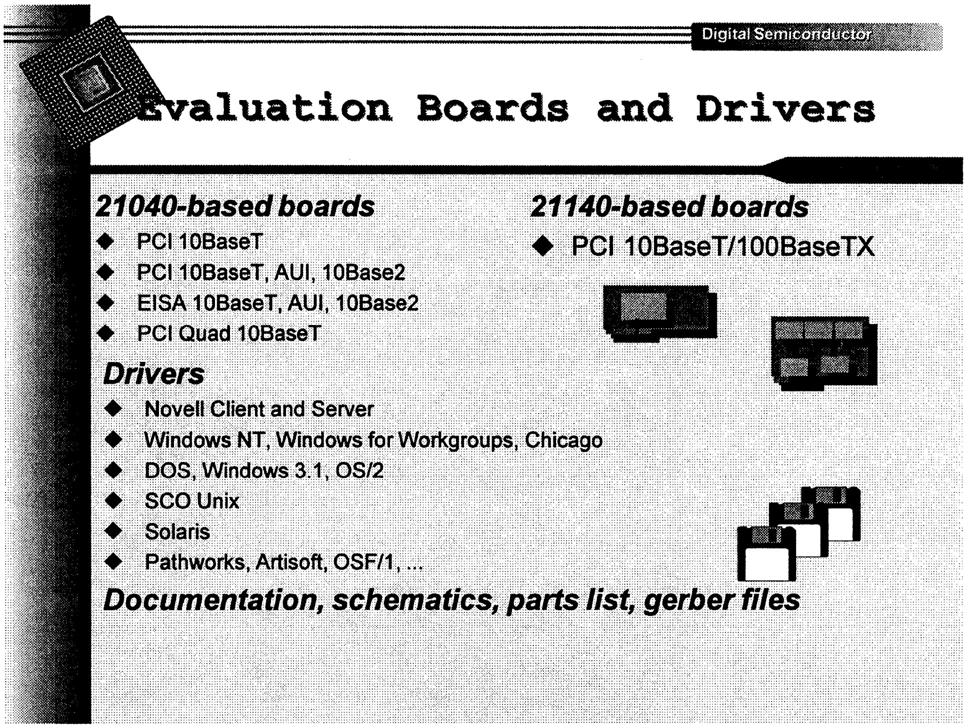
- ◆ Ideal for servers
- ◆ Single 1/2 slot PCI adapter card
- ◆ Low CPU utilization => Many available cycles for running applications
- ◆ 4 port 21040/41 adapters with Full Duplex provides 40Mbps for receive and 40Mbps for transmit



Digital Semiconductor

Follow-on Products

- ◆ 21041
  - Follow-on and driver compatibile with 21040
  - Boot ROM support (up to 256 Flash Memory)
  - NWay Full Duplex Autodetection
  - Support for 3.3V and 5V PCI environments
  - Power-down mode
  - Write-able IEEE address support
  - Real-time clock
  - Samples now, production in March
- ◆ 21140 follow-ons
  - Enhancements similar to 21041
  - External CAM support
  - Motherboard focus



Digital Semiconductor

Evaluation Boards and Drivers

21040-based boards

- PCI 10BaseT
- PCI 10BaseT, AUI, 10Base2
- EISA 10BaseT, AUI, 10Base2
- PCI Quad 10BaseT

21140-based boards

- PCI 10BaseT/100BaseTX

Drivers

- Novell Client and Server
- Windows NT, Windows for Workgroups, Chicago
- DOS, Windows 3.1, OS/2
- SCO Unix
- Solaris
- Pathworks, Artisoft, OSF/1, ...

Documentation, schematics, parts list, gerber files

Directions in PCI LAN Adapter Design

Jeff Stockdale
Director of Engineering
Cogent Data Technologies, Inc.
P.O. Box 926
175 West St.
Friday Harbor, WA 98250
(800) 426-4368/(360) 378-2929/2882 (fax)
jeff@cogentdata.com

Cogent Data's LAN adapter product line is based on the Digital Semiconductor LAN chips. We plan to extend our line to cover Fast Ethernet as well as other emerging high-speed networks. The PCI bus is especially well-suited to the design of fast servers because of its high bandwidth, secondary bus capability behind PCI bridges, and ability to support multiport adapters.

Cogent is a leader in the design and manufacture of state-of-the-art Ethernet and Fast Ethernet adapters. Cogent's latest product announcements include the PCI Quartet adapters in both 10BaseT and 100BaseTX versions which provide four independent Ethernet segments on a single PCI adapter.

Jeff is presently the Director of Engineering at Cogent, responsible for all hardware and software design activities. He has been with the company for two years and has been managing the development group for the past year. Prior to Cogent, Jeff was employed by Attachmate Corporation as a Product Manager and Senior Hardware Engineer. Jeff has 12 years of communication industry experience and has an Electrical Engineering degree from Washington State University.

Developing Network Interface Cards for the PCI Bus

Jim Schooler
Senior Product Manager for Adapter Products
Racal InterLan, Inc.
60 Codman Hill Road
Boxborough, MA 01721
(508) 881-2308/263-8655 (fax)
schooler@interlan.com

This discussion will center on the ups and downs of developing network interface cards for the PCI bus. This includes the differing operating systems (and network operating systems) and how they operate (or fail to operate) with the PCI bus. Attention will be paid to design, development, and testing for compliance to the PCI bus.

LOW-COST, HIGH PERFORMANCE 10/100 PCI ADAPTER DESIGN FOR 100VG-ANYLAN

Lisa Piper
AT&T Microelectronics
555 Union Boulevard
Allentown, PA 18103

ABSTRACT

One of the primary motivations for upgrading computer equipment is the desire for more computer power. This enables faster response times and it enables new applications like multimedia. It is clear that next generation high speed computers have migrated to the PCI architecture as one means of achieving greater computing power. The focus of this paper is on how LAN technologies are evolving to meet the local networking needs of these higher bandwidth networked computers. 100VG-AnyLAN is discussed in detail, including its cost advantages and technical characteristics. An example PCI adapter card design is also discussed.

INTRODUCTION

More powerful computing capabilities has led not only to faster response times but also has enabled new multimedia applications. There is now a movement towards higher speed networking. Several alternatives exist, however, this paper proposes that 100VG-AnyLAN technology is the way to go for the near future. Next generation networking solutions are compared, the advantage of using 100VG-AnyLAN is discussed, a technical overview of 100VG-AnyLAN is provided, and an example 100VG-AnyLAN PCI adapter card design is shown.

HIGH SPEED NETWORK COMPUTING SOLUTIONS

High speed networking will make available more bandwidth. This could mean more users on a network. But more likely it means being able to use the network to do applications like imaging that require that large files can be transmitted across the network in a timely manner. And for multimedia, low latency and guaranteed bandwidth are also required. Typical bandwidth needs for multimedia are from 384 kbits/s to 2 Mbits/s.

Criteria for Selecting a New Network

1. *Price/Performance*: Improve performance for a modest premium.
2. *Compatibility*: Preserve as much of the existing network as possible. This includes the infrastructure (i.e. bridges, routers, hubs, adapter cards), cabling, network topologies, network operating system software(NOS), and network management capabilities.
3. *Simple, Low Cost Migration*: Users should be able to evolve to the new network with minimal disruption and low migration costs.
4. *Provisions for Emerging Applications*: In today's environment, that implies low latency and guaranteed bandwidth for multimedia..

There are four primary buyer concerns when upgrading network capabilities. Buyers want better performance for a modest premium. This not only includes bandwidth, but also provisions for emerging applications. In today's environment, that includes low latency and guaranteed bandwidth. Minimal change is desired to minimize cost and to make the migration simpler. This means preserve as much of the existing network as possible, including the infrastructure (i.e. bridges, routers, hubs, adapter cards), cabling, network topologies, network operating system software(NOS), and network management capabilities. Compatibility and low cost migration go hand in hand. The more compatible the two technologies are, the easier it is to evolve in steps, thus spreading out the cost and conversion time incrementally.

The predominant LAN in use today is clearly Ethernet, and the biggest reason is that it operates over category 3 UTP cabling since most existing wiring is Category 3 UTP. Therefore, it makes sense to judge compatibility and cost by a comparison with Ethernet. Table 1 lists alternative 100 Mbits/s technologies.

Table 1. - High Speed Alternatives to the Desktop

Alternative	Data Rate (Mbits/s)	Data Grade UTP (Cat 5)	Voice Grade UTP (Cat 3)	Multimedia Support	Ethernet Topology	Rel. Cost 1=low, 6=high	Network Span
Ethernet	10	100 m	Yes	No	Yes	1	1100 m
100VG-AnyLAN	100	150 m	100 m	Yes	Yes	2	2200 m
Token Ring	4/16	40-100 m	40 m	No	No	2	Large
100Base-T4	100		Planned	no	no	2	210 m
100Base--TX		Yes				2	
CDDI	100	Yes	No	Limited	No	4	2200 m
ATM	25, 50 155	Planned Planned	Planned No	Yes Yes	No No	6	Large

Ethernet and Token Ring are included in Table 1 for base comparisons. One additional comment on ATM is that today there are still a lot of standards issues, there are not clear choices, and the technology is still very expensive. While ATM has a lot of potential for the future, it is not likely to be installed to the desktop today. It is clear from the table that 100VG-AnyLAN will provide performance superior to that of Ethernet and will support emerging applications.

100VG-ANYLAN OVERVIEW

This section provides an overview of 100VG-AnyLAN technology with respect to how it addresses buyer concerns of compatibility, simple low cost migration, and the ability to support emerging applications. The next section will address price/performance by showing the simplicity of an adapter card design.

Compatibility of 100VG-AnyLAN

There are two basic characteristics of 100VG-AnyLAN that account the most for making 100VG-AnyLAN very compatible with Ethernet. The first is that 100VG-AnyLAN uses quartet signaling which allows for running over UTP category 3 cabling. Quartet signaling uses a four level transmission scheme and transmits a scrambled 5B/6B NRZ encoded data across four pairs of category 3, 4, or 5 UTP at a rate of 30 Mbits/s. Refer to Figure 1. The transmission rate is only slightly greater than that of Manchester encoded Ethernet, so EMI will be about the same; the scrambling and encoding have resulted in performance improvements(transmission distances possible) over Ethernet. Performance has proven to exceed those of Ethernet and all the same Ethernet topologies are possible. This means the customer's wiring will not have to change when migrating from Ethernet.

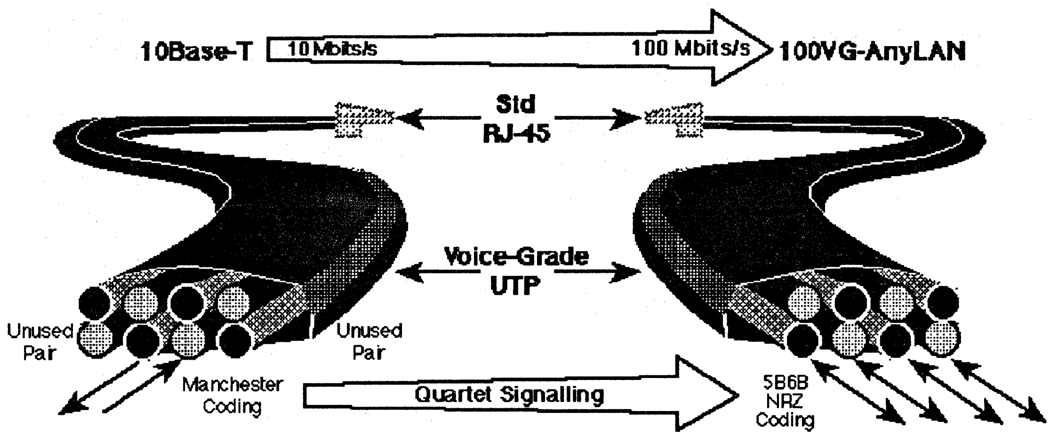


Figure 1. 100VG-AnyLAN Quartet Signaling

The second characteristic that makes 100VG-AnyLAN so compatible is that it uses the same frame format as in Ethernet. In addition, 100VG-AnyLAN can alternatively use a Token Ring frame format. Using the same frame format enables the buyer to still use existing network operation system software drivers and bridges and routers. This makes 100VG-AnyLAN very compatible with both Token Ring and Ethernet networks.

Migrating to 100VG-AnyLAN

Migrating to 100VG-AnyLAN is very easy. It is only necessary to change out the adapter card and the hub. Most adapter cards are being designed so they can be used on Ethernet or 100VG-AnyLAN networks. Buyers can purchase a 10/100 card and use the Ethernet mode until the hub is changed out. These new adapters even autoselect 100VG-AnyLAN mode so the user does not have to do anything to change once the 10/100 card is installed.

Most hubs are designed to provide either an Ethernet port or a Token Ring port that can be connected to the old network. This means the buyer can use existing bridges and routers. Commercially available repeater IC's , the ATT2R01 for example, provide filtering features to simplify this bridge port design and they provide the same network management statistics and security provisions that more advanced Ethernet users are accustomed to.

Application Flexibility of 100VG-AnyLAN

Ethernet and most existing LANs are designed primarily for data transmission. They cannot guarantee bandwidth or bound latencies. This is becoming an issue with emerging applications like multimedia.

100VG-AnyLAN uses a hub-centric demand priority protocol. Packets are directed on the fly from the source to the destination and there exist two levels of priority. Stated very simply, the source "requests" to transmit. The hub addresses these "requests" using two round robins, one for each level of priority. High priority packets are "given permission to transmit" first. There are no collisions in 100VG-AnyLAN so transmission is much more efficient than Ethernet. Latencies can be bounded by limiting the size of the round robin and the number of stations allowed to transmit at high priority. "Requests" and other control information is achieved through the transmission of low frequency tones.

DESIGN OF A 100VG-ANYLAN PCI ADAPTER CARD

This section shows that it is relatively easy and cost effective to develop 100VG-AnyLAN products by discussing an example 100VG-AnyLAN PCI adapter card design and showing the cost of it versus the cost of a similar Ethernet card.

Relative to the initial Ethernet IC offerings, 100VG-AnyLAN IC's provide very high levels of integration and maintain the features that have become standard over time for Ethernet. Figure 2 shows a high level block diagram of the example PCI adapter card.

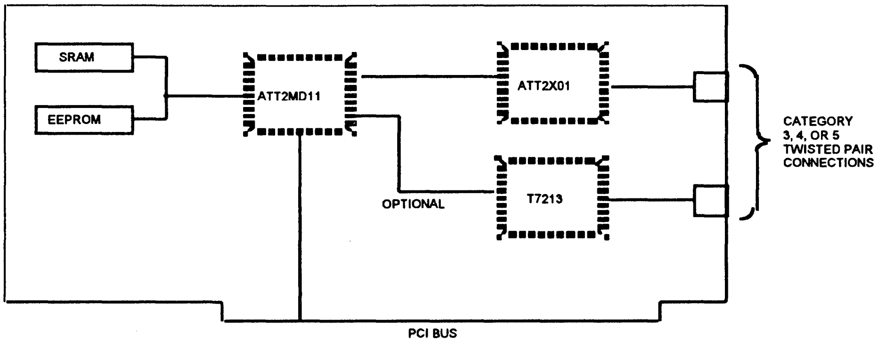


Figure 2. 100VG-AnyLAN PCI Adapter Card

There are two IC's in the design that are specifically for 100VG-AnyLAN. These are the ATT2X01 100VG-AnyLAN transceiver and the ATT2MD11 MAC and PCI System Interface device. An Ethernet transceiver, the T7213 for example, can be added to complete the Ethernet offering.
The best way to understand the card design is to understand the IC's in it.

ATT2X01 Transceiver for 100VG-AnyLAN

The ATT2X01 provides the transceiver functions needed in a 100VG-AnyLAN Category 3, 4 or 5 solution. It enables 100 Mbits/s transmissions over 10Base-T wiring installations (bundled or unbundled). A block diagram of the device is shown in Figure 3.

The ATT2X01 provides the four transceivers necessary to connect to one category 3 interface. The receiver circuitry detects energy, provides automatic gain control, adaptive equalization, phase locking, and data justification. On-chip drivers minimize component count and facilitate FCC compliance by simplifying board layout. Transmit and receive line state machines and associated tone detectors and tone generators support the two levels of priority needed for bounded latency networking (for multimedia and other time sensitive applications). The ATT2X01 can be used in station or hub designs.

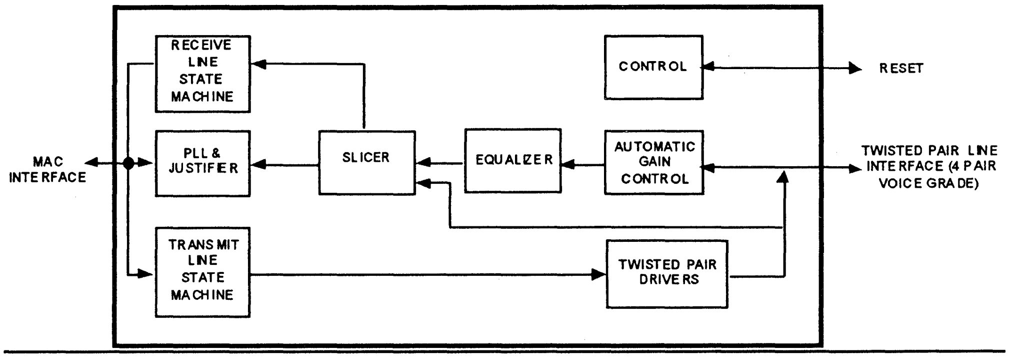

RECEIVE LINE STATE MACHINE
CONTROL
RESET
MAC INTERFACE
PLL & JUSTIFIER
SLICER
EQUALIZER
AUTOMATIC GAIN CONTROL
TWISTED PAIR LINE INTERFACE (4 PAIR VOICE GRADE)
TRANSMIT LINE STATE MACHINE
TWISTED PAIR DRIVERS

Figure 3. ATT2X01 100VG-AnyLAN Transceiver Block Diagram

ATT2MD11 MAC and PCI System Interface Device

The ATT2MD11 provides all the MAC, memory management, and PCI bus interface logic in one chip. Furthermore, two MACs are available - one for Ethernet and one for 100VG-AnyLAN. This facilitates the design of 10/100 products which makes it easier for the buyer to make gradual transitions and enables the design of portable test equipment that can be used across networks. A block diagram of the ATT2MD11 is shown in Figure 4.

In addition to the two MACs, there is an associated autoselect circuit. This circuitry will determine if a 10Base-T link exists. If it does, the Ethernet MAC will be selected automatically. Otherwise the 100VG-AnyLAN MAC will be selected. This way the end user will not have to do anything when a 100VG-AnyLAN hub is added; the transition will happen automatically.

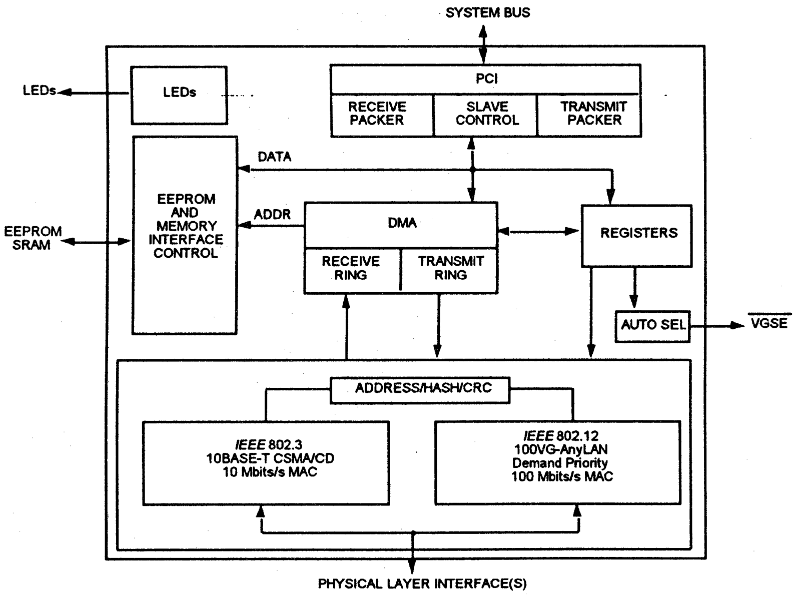

SYSTEM BUS
PCI
RECEIVE PACKER
SLAVE CONTROL
TRANSMIT PACKER
LEDs
LEDs
DATA
ADDR
EEPROM AND MEMORY INTERFACE CONTROL
EEPROM SRAM
DMA
RECEIVE RING
TRANSMIT RING
REGISTERS
AUTO SEL
VGSE
ADDRESS/HASH/CRC
IEEE 802.3 10BASE-T CSMA/CD 10 Mbits/s MAC
IEEE 802.12 100VG-AnyLAN Demand Priority 100 Mbits/s MAC
PHYSICAL LAYER INTERFACE(S)

Figure 4. ATT2MD11 MAC and PCI Interface Block Diagram

The ATT2MD11 also provides the PCI bus interface, the memory manager, and glueless interfaces to an optional EEPROM and to memory. The 32 bit PCI interface is designed to connect directly to the PCI connector and will support PCI slave modes at maximum bus speeds. Slave mode ensures broad compatibility and minimal software overhead.

The memory manager is the same as that of the ATT2MD01 MAC and ISA/EISA interface device. This allows the software data structures to be maintained across PCI slave, EISA slave and ISA slave implementations. The memory manager supports up to 128 Kbytes of local memory to ensure host bus efficiency. Also, byte, word, or double word transfer capability optimizes performance.

The EEPROM is used to store configuration information. It can be omitted for motherboard applications or other similar applications where a EEPROM exists someplace else on the board. The ATT2MD11 also provides the typical adapter card LED activity and status indicators.

To summarize, the 100VG-AnyLAN adapter card consists of the transceiver, the MAC and system interface chip, 128 Kbytes of SRAM, and a EEPROM. The transceiver will require magnetics and some miscellaneous resistors and capacitors. Two oscillators are also needed by the ATT2MD11.

In addition to the high integration of functions in silicon, development of 100VG-AnyLAN products is facilitated by having the IEEE 802.12 Standard in place, development tools are available, and interoperability tests are in place at University of New Hampshire. Together, these measures ensure the buyer a smooth migration and developers a quick time to market.

Adapter Card Cost Comparison to 10Base-T

Now that the design of a typical adapter card is understood, we can compare the cost of a 10Base-T PCI adapter card to a 100VG-AnyLAN adapter card. Table 2 shows the major components and 1994 estimated costs of each design. The main differences are in the cost of the 100VG-AnyLAN controller and transceiver. Some additional cost will also occur in 100VG-AnyLAN; this is a result of special crystals and precision capacitors, and some accounting for a slightly more complex assembly and test process since 10Base-T designs typically also integrate the transceiver. These differences will diminish quickly as 100VG-AnyLAN is more widely deployed.

Table 2 - Cost Comparison of 10Base-T PCI to 100VG-AnyLAN PCI

Component (100K Volume) (1994)	10Base-T	10Base-T, 100VG-AnyLAN
Controller/Xcvrs	$25.00	$45.00
SRAM	$11.00	$11.00
Assembly/Test	$8.00	$10.00
Magnetics	$3.00	$5.00
PC Board	$5.00	$8.00
Miscellaneous	$1.75	$8.00
Total	$53.75	$87.00

As is shown from Table 2, there is less than a two to one differential in cost for 10 Mbits/s 10Base-T and a combination 10Base-T/100 Mbits/s 100VG-AnyLAN card. This small increase in cost buys new application potentials and a tenfold increase in performance, probably more since 100VG-AnyLAN is much more efficient than 10Base-T because there are no collisions.

SUMMARY

100VG-AnyLAN provides a cost effective migration path for both Ethernet and Token Ring users. Use of 100VG-AnyLAN preserves the existing infrastructure, including bridges and routers, network topologies, cabling, and network operating system and network management software. 100VG-AnyLAN provides a substantial performance improvement for a modest premium over Ethernet and an even lower premium over Token Ring. It provides higher speed and low latency, guaranteed bandwidth so it can support emerging multimedia applications. And migrating to 100VG-AnyLAN is as simple as replacing the hub and the adapter card.

A design for a 100VG-AnyLAN adapter card was presented and shown to be very cost competitive relative to Ethernet. Highly integrated silicon is readily available. The 802.12 proposed standard is in place and is expected to be adopted soon. Test equipment exists and interoperability labs are set up. Equipment is readily available from multiple vendors today.

The book, Planning and Designing High Speed Networks Using 100VG-AnyLAN provides an excellent tutorial on 100VG-AnyLAN. To learn more about the ATT2X01 ATT2MD11, ATT2R01, or ATT2MD01, contact AT&T Microelectronics Customer Response Center at 1-800-372-2447, Department P; in Canada, 1-800-553-2448, Department P; for customers outside of the U.S., fax +1-610-712-4106. AT&T Microelectronics offers both repeater and adapter card demonstration systems that can be used to learn about the devices and about 100VG-AnyLAN.

AUTHOR BIOGRAPHY

Lisa J. Piper is Manager of the Application Engineering group that supports AT&T Microelectronics LAN IC Products. She joined AT&T in 1981 and has experience in LAN design, VLSI design, and ISDN terminal design. Lisa has a B.S.E.E. from Purdue University and a M.S.E.E. from Ohio State University.

REFERENCES

Costa, Janis Furtek. Planning and Designing High Speed Networks Using 100VG-AnyLAN. Englewood Cliffs, NJ: Prentice-Hall, 1994.

Expanding PCI Bus for High Speed Communication in SuperComputing Environment

N Gopal Reddy and Shreyas Shah*

Abstract

PCI (Peripheral Component Interconnect) is a processor independent local bus. It has received wide acceptance from Computer Industry. In this paper we have proposed some changes to the existing PCI standard for the following reasons.

1. Enhance the Bus performance.

2. To get wider acceptance for a longer duration.

1 Introduction

We have briefly described PCI bus protocols, DS (Data Strobe) link and Cluster Computing in the following sections.

1.1 PCI Bus :

PCI is a 32/64 bit processor independent Local Bus. It is a synchronous Bus with operating frequency in the range of DC to 33 Mhz. PCI is based on reflected wave technology, hence it converts ringing problem of high speed bus into an advantage. PCI compliant devices have to drive the Bus for half the TTL voltage level unlike other standard Buses. Hence the required driving capability of PCI compliant devices are halved, reduces power consumption. It supports Data and Address stepping which reduces ground bounce and allow to use weak buffers. It also supports auto configuration, multiple voltage signalling environment and variable length linear and toggle bursting.

1.2 DS Link :

DS Link[1] is a high speed full-duplex serial communication link.
It consists of four wires, two in each direction, one carrying data and another carrying strobe, hence the term DS-links (data-strobe). Various Media recommended for DS link are PCB tracks, copper cable or single/multimode fibre cable. Each link can operate at upto 100 Mbits/sec, providing a bidirectional bandwidth of 19Mbytes/sec. The link protocol supports virtual channels, dynamic message routing and provides a high data bandwidth. The DS-link protocols are part of the proposed IEEE standard for Heterogeneous Interconnect (IEEE P1355). This standard proposes simple scalable point to point connection schemes ranging from 100 Mbaud to 1 Gbaud speeds. DS link has separate Data and Strobe lines. Strobe is generated by exclusive-or operation on Data and clock. Hence clock can be retrived from Data and Strobe at the receiver without any significant effort.
DS link is based on packet transmission. Each packet has header which provides routing information.

*The Authors are with Centre for Development of Advanced Computing (C-DAC), India

[1] DS Link is a Trade mark of SGS-Thomson, U.K.

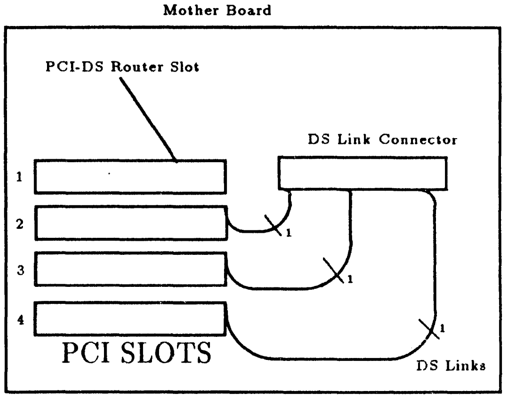


Fig.1 Mother Board with PCI-DS Router Slot, Reserved Pins for one DS Link/slot One DS Link from each PCI slot to Router Slot, Edge Connector for DS Links on PCI-DS Router slot for external Interface, PCI-DS Router with PCI Bus Interface

1.3 Cluster Computing:

Cluster computing is a technique to run various applications on different workstations connected through High speed InterConnect(HIC). This creates a virtual parallel environment. DS Link has many advantages compared to conventional interconnect schemes for cluster computing applications. Following are some of the advantages of DS Link in Cluster Computing application.

1. High Performance Packet switching
2. Availability of fully configurable wormhole Asynchronous Packet routers
3. Low latency communication protocols required in efficient Communication for computing
4. Availability of routers for Group-Adaptive and Universal Routing to minimize hot spots
5. Low cost interconnection per port

2 Suggestions:

2.1 Incorporating DS Link in PCI Bus:

This can be performed by incorporating DS links as part of the PCI bus signals. This can be achieved by two different ways.

- Using reserved pins of PCI bus.
- Adding extra connector with PCI connector to support multiple DS Links per slot.

2.1.1 First Approach :

In present 32 bit PCI bus connector, there are only 6 reserved pins, hence it is possible to go for one DS Link per slot[2]. The reserved pins assigned for DS Link in each slot can be connected to the router. This router can be either integrated on the motherboard or it can be on the PCI-DS router slot[3]. There are two approaches to achieve this.

1. To have one PCI-DS router slot and one DS Link support (use reserved pins) for each slot on PCI. Refer Figure No. 1
2. To integrate DS router on the mother board and use reserved pins of PCI connector for DS Link. Refer Figure No. 2

2.1.2 Second Approach :

The first approach is not suitable for the application that demand high communication bandwidth e.g. Cluster Computing. So it is necessary to have multiple DS Links per device. There are two approaches to achieve this.

1. In each PCI based system, there will be one PCI-DS router slot and one or more number of PCI-DS agent slots[4]. Refer Figure No. 3
2. To integrate DS router on the mother board and have multiple PCI-DS agent slots. In this approach, there will not be any PCI-DS router slot. Refer Figure No. 4

We recommend to have an integrated DS router on the mother board and DS connector for accepting upto four DS Links with each PCI slot in the system (second approach: item no. 2). In this approach, the router chip with PCI interface

[2] Each DS Link require four lines

[3] PCI-DS Router slot means one PCI connector with a DS Link connector to accept all the DS Links from PCI-DS agents

[4] PCI-DS agent slot means PCI connector with small DS link connector to accept four DS links from the agent in that slot. These Links will be routed to PCI-DS router slot

can be optional. There can be separate DS connectors mounted on the motherboard that accepts DS Links meant for external communication on the router. This enables the system to communicate to other workstations/high performance systems with DS link support. Depending on the requirement, user can plug in the DS router chip available from standard sources.
This scheme will have following advantages over other schemes.

1. Theoritically, PCI can support 10 Loads. In the practical implementation of PCI Bus system, the maximum number of PCI slots can be upto four (Because each slot is equivalent to two loads). So by using integrated router, the load capacitance on PCI Bus can be significantly reduced.

2. Integrated devices on the mother board can also become members on the DS Link network.

3. Integrated DS Router on the motherboard will increase the reliability of the system.

4. This approach is also cost effective.

In the above mentioned schemes, the devices on one PCI bus can communicate to each other through very high speed parallel bus (PCI) as well as high speed serial link (DS Link). Since the DS link router is an intelligent asynchronous crossbar packet switch, it is possible for multiple devices on the PCI Bus to communicate simultaneously through DS links. This will help in considerably reducing the traffic on PCI Bus and enhance the communication bandwidth between PCI agents.
Machines with DS Link support on the motherboard, can be connected to each other through a simple central crossbar switch. Any PCI agent with DS Link support can communicate with any other agent of the system which is connected to the DS network without intervention of PCI Bus

- Incorporating DS Link in PCI Bus helps in exploiting the inherent advantages of DS Link for efficient Cluster Computing. There are devices available for interfacing parallel buses to DS Link (DS Link Adaptor).[5] and routing the information among machines (DS Router) [6] connected through DS Links[7].

- PCI bus agents with DS Link support can concurrently communicate through PCI bus as well as very high speed DS link.

2.2 Upgrade Path from 32/64 bit to 128/256 bit PCI Bus:

At present all latest processors support 128/256 bit Memory data path width. Since PCI is a Processor independent bus, there should be some mechanism to allow users to extend the PCI Bus width to 128/256 bit and hence increase the bandwidth by two to four fold.
We suggest that one can implement above mentioned as follows. The initiator will find whether target supports 128/256 bit data bus by using special cycle. If so, it will communicate in the 128/256 bit mode else it communicates as per standard PCI 32/64 bit protocol. The extra connector for this enhanced PCI has to be standardized by PCI SIG (Special Interest Group).

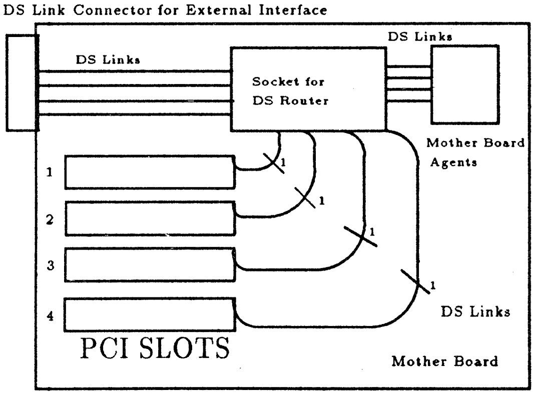


Fig. 2 Reserved Pins of PCI used for DS Link DS Router on MotherBoard

[5] ST C101: DS Link Adaptor
[6] ST C104: DS Router
[7] ST C101 and ST C104 are available from SGS-Thomson

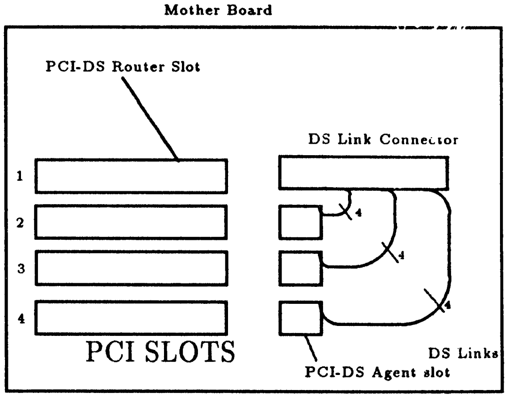


Fig.3 Mother board with PCI-DS Router Slot, PCI-DS Agents Edge Connector for DS Links on PCI-DS Router Slot card for External Interface, Four Links from each PCI-DS Agent to PCI-DS Router Slot

2.3 Dynamic configuration support in PCI Bus Specifications:

In the PCI bus there is no way that PCI master can initiate a fresh configuration cycle. We suggest that there should be atleast one line that informs default master to initiate a new configuration cycle. There are many applications where the PCI cards need to be reconfigured for some parameters like min-gnt and max-latency. We call this a dynamic configuration. This allows to meet certain timing parameters for specific application. So in those kind of applications, first the card will have some default values in configuration registers, these will be changed by default master according to some rules. If those parameter values are not suitable for application then card will change those default configuration parameters and restart a configuration cycle to meet timing requirements of that application.
So this is as good as some network protocol where network card will find the negotiated parameter values and if it is not suitable for the application, it will restart negotiating for parameters to meet exact timing performance of a particular application. Preliminary work in this direction is being carried out and will be reported elsewhere.

Conclusion

From the above mentioned suggestions, we conclude that

1. PCI Bus based systems have to be modified to support High speed DS Link interconnect; this will help in the design and development of very high performance (Tera flops) parallel machines.

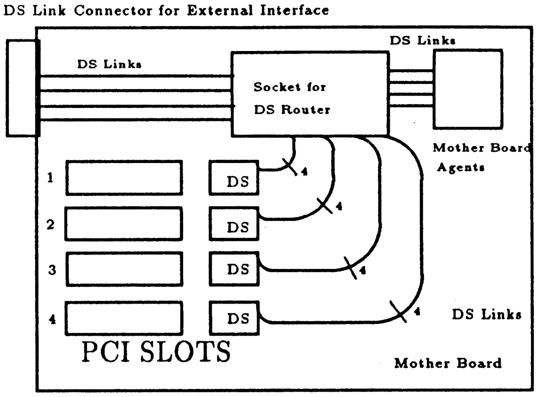


Fig. 4 Extra Connector for DS Link On every Slot, 4 DS Links/slot DS Router on MotherBoard

2. Upgrading PCI Bus to 128/256 bit will support latest developments in processor architectures. Hence the system performance may increase by two to four fold.

3. Incorporating dynamic configuration feature in PCI, will efficiently serve many real time applications.

Acknowledgement

The authors would like to thank Mr Yogindra Abhyankar for his useful criticisms and suggestions. His detailed comments have greatly improved the quality of the paper. The authors also thank Dr. Colin WhiteBy-Strevens of SGS-Thomson, U.K. for his valuable suggestions.

References

[1] PCI BUS specification Rev 2.0
[2] STC101 data sheet from SGS-Thomson, U.K.
[3] STC104 data sheet from SGS-Thomson, U.K.

Contact Address:

N Gopal Reddy, Shreyas Shah

Centre for Development of Advanced Computing,
Pune University Campus, Pune 411 007, India.

Phone : (91) (212) 332531
Fax : (91) (212) 337551

E-mail : gopal@parcom.ernet.in
shreyas@parcom.ernet.in

CARDBUS: A KEY TO PERSONALIZED PORTABLE COMPUTING

Claude A. Cruz
National Semiconductor Corporation
333 Western Avenue, M/S 10-26
South Portland, ME 04106

ABSTRACT

Portable computers are proliferating at an ever-growing rate. These devices now fill out a computing-platform taxonomy that include laptops, notebooks, sub-notebooks, personal digital assistants (PDAs) and personal communication systems (PCS). All of these devices share certain common constraints, such as the need for extreme power efficiency while delivering computing capabilities that rival those of full-size desktop systems.

The PCI bus supports these machines with its wide data/address path (32+ bits), its high operating frequency (33+ MHz), its ability to operate at either 5v or at a power-saving 3.3v, and its numerous advanced features (e.g. cache-coherency mechanism, bus-locking, multi-master capability, etc.).

CardBus is a high-end extension of the popular PCMCIA standard for dynamically-installable system options. Architecturally, CardBus is a close cousin of PCI, making pairing of these two interfaces especially attractive. Through its "hot insertion" capability and associated configuration software, CardBus allows functions operating at local-bus speeds to be attached to a system as needed. This makes it possible to "tailor" computers to individuals' needs and preferences with unprecedented ease.

This paper explores some of the novel ways in which the conjunction of PCI with CardBus will reshape computing in both the portable environment and on the desktop.

CARDBUS AS A PCI DERIVATIVE

Because CardBus is intended to serve as a high-performance system reconfiguration mechanism, it is patterned closely on the PCI bus. There are five major areas in which CardBus extends and/or complements PCI. To better understand this complementarity, we will sketch the major characteristics of the PCI bus and of PCMCIA/CardBus in the following areas:

- Performance at local-bus speeds;
- Bus-master capability;
- Hot-insertion/dynamic configuration capability;
- Small physical form-factor card and connector;
- Emphasis on minimizing power dissipation.

PCI Overview

The PCI bus standard was developed to specify a uniform local-bus medium that can be used as a system back-bone across a range of computer performance levels. PCI enables the design of systems which may contain several system buses, to enable concurrent transfers between various functional units of the machine. The PCI bus is especially efficient in executing burst (i.e. multi-datum) transfers, for which bus-arbitration and addressing overhead are minimized.

PCI systems can accommodate a number of master devices, as well as slave cards. The PCI standard includes a protocol for bus arbitration, as well as stipulating constraints on bus acquisition utilization which are intended to minimize bus latency and ensure equitable partitioning of bus bandwidth across multiple masters. PCI includes a "LOCK" protocol by which masters can obtain exclusive use of the bus for some period.

In a PCI system, cards are assumed to be installed in the system motherboard before power is applied. Then, during system boot-up, the PCI configuration space is used to allocate shared resources such as interrupt levels to specific PCI agents. The PCI configuration process is mediated by low-level software (typically in device drivers), and is hardware-specific. PCI cards cannot be installed into or removed from the system after boot-up.

The PCI standard defines a 120-pin connector (for a 32-bit bus) which can be used to connect a card directly to a PCI bus. This connector is used with a fairly large-format card which is intended for use in a desktop computer.

The PCI standard focuses on providing adequate hardware support for system power-reduction, and codifies operation at 3.3v as well as at 5v. In addition, the standard allows clock-gating and/or frequency change to lower card power consumption in a controllable fashion.

PCMCIA/CardBus Overview

Since its inception in the 1980's, the PCMCIA (Personal Computer Memory Card International Association) standard has provided a means for adding functions to operating computer systems. Initially, PCMCIA was used to add ROM, Flash and SRAM memory to a computer. These "Revision 1" ("R1") cards were soon followed by a variety of I/O ("R2") device adapters.

CardBus was defined primarily to provide higher performance than can be achieved using PCMCIA R1 or R2, which utilize an 8-bit or 16-bit interface operating at ISA bus speeds (8 MHz). In contrast, CardBus provides a 32-bit multiplexed address/data path which operates at PCI local-bus speeds of up to 33 MHz. CardBus accomplishes this by adopting the synchronous burst-transfer orientation of PCI, as well as a bus protocol which is essentially identical to that of PCI. These similarities makes it especially easy to link CardBus with PCI (although CardBus is also usable with other system buses, such as ISA or EISA). Note that the CardBus standard is a superset of the PCMCIA standard, allowing existing PCMCIA cards to be used in CardBus sockets.

Besides its PCI-like data rate, CardBus devices are capable of acting as system-bus masters; that is, they can assume control of the system bus(ses) to effect data transfers. This contrasts with PCMCIA R1 and R2 devices, which can only act as slaves to system-resident master devices. The CardBus master capability opens a path to novel types of "intelligent" system adapters, as is discussed later in this paper.

Both PCMCIA and CardBus view dynamic system reconfiguration as a primary functional requirement. This reflects the fact that these interfaces must allow hardware such as modem or Flash memory cards to be shared by simply porting them from one system to another. This resource mobility allows users to carry their work environment with them, in the form of "personalizable" hardware adapters.

The PCMCIA and CardBus interfaces allow an I/O adapter or a memory module to be added to or removed from an operating computer without disrupting the system's operation. Moreover, both PCMCIA and CardBus standards define a resource-configuration software architecture which allows high-level configuration of system resources whenever cards are added to or removed from a system.

"Card Services" consists of a generic high-level mechanism for describing a card's capabilities and system-resource needs. Card Services is based on a card-description "Meta-format" which has been gracefully extended from PCMCIA to CardBus.

"Socket Services" is a hardware-dependent software layer which intervenes between Card Services and a particular PCMCIA or CardBus host-bus bridge. Socket Services translates Card Services' generic card status and command transactions into the accesses which are required by a particular PCMCIA or CardBus controller IC.

The PCMCIA standard defines a compact 68-pin connector and several small, thin card form-factors. CardBus cards share PCMCIA's mechanical design, as well as a 68-pin connector similar to that used by PCMCIA cards. In the CardBus connector, a special shielding shroud is used to provide enough signal integrity to operate the bus at up to 33 MHz. When used with the prescribed CardBus controlled edge-rate buffers, the grounded shield ensures that ground-bounce does not corrupt signals.

Both PCMCIA and CardBus place heavy emphasis on supporting card power reduction. Card Services defines Meta-format "tuples" which describe a card's power requirements and options. This information can be used to control the power (Vcc and Vpp) supplied to a card socket. In addition, CardBus includes PCI's CLKRUN# signal and its associated protocol for controlling clocking to a card.

Figure 1 Summarizes the similarities and complementarities between PCI and CardBus.

CARDBUS APPLICATIONS IN PCI SYSTEMS

The preceding section suggests that CardBus can serve as a performance-matched means for adding hot-insertion capability to a PCI-based system. Moreover, the CardBus and PCI interfaces are (intentionally) so similar that they can easily and effectively be used together. Finally, the small CardBus form-factor and power-reduction capabilities make CardBus an ideal adjunct to PCI in constructing high-performance portable systems.

	PCI	CardBus
PERFORMANCE		
Data/Address Width (bits)	32/64	32
Max. Clock Rate (MHz)	33/66	33
Peak Transfer Rate (MB/sec)	132/264	132
Bus-Master Capability	YES	YES
CONFIGURATION		
Hot-Insertion Support	NO	YES
Boot-up Configuration Support	YES	YES
Dynamic (In-Operation) Configuration Support	NO	YES
Configuration-Software Level	Low (Device Driver)	High (Card/Socket Servcs)
POWER MANAGEMENT		
Operating Voltage(s)	5v/3.3v	3.3v Only
Card-Clocking Hardware Support	NO	YES (CLKRUN)
MECHANICAL DESIGN		
Card Form-Factor	Desktop (ISA-Like)	Portable (Credit-Card-Size)
Connector Type	120-Pin Unshielded	68-Pin Shielded
Card Bridge Hardware Required	NO	YES

Figure 1: PCI/CardBus Comparison

To flesh out this proposed use of CardBus, two other PCI derivatives need to be put into perspective: PCI Small Form-Factor (PCI SFF), and PCI Mobile. All three of these PCI variants are well-suited for use in portable systems, and indeed are most often used in portable systems.

PCI SFF is basically a repackaging of standard PCI cards to provide a smaller card format which is more compatible with physically-compact PCI-based systems. PCI SFF cards can be directly connected to a PCI bus, without intervening bridge hardware. PCI SFF cards are not "hot-insertable"; that is, they cannot be inserted into or withdrawn from a powered socket, as is required to support dynamic system reconfiguration. In addition, the PCI SFF standard shares PCI's configuration mechanism, and lacks definition of the high-level software configuration mechanisms provided by PCMCIA and CardBus.

The PCI Mobile standard is a variation of the PCI standard which focuses on power-reduction issues (which are of central importance in battery-driven portable systems). Like PCI SFF, though, this is a hardware-oriented standard which does not provide hot-insertion capability or high-level configuration software.

Both PCI SFF and PCI Mobile are candidates for use in systems which do not require hot-insertion or dynamic reconfiguration. Typical applications might include adding an internal LAN adapter to a portable system. This type of PCI card can be simply and inexpensively connected directly to the platform's PCI bus(es). CardBus is a better choice for adding shareable high-performance adapters to a portable system. CardBus performance is equivalent to that of PCI SFF or PCI Mobile cards. However, the versatile system reconfiguration capability of requires support from CardBus-to-PCI bridge controllers and more-complex (though standardized) system software.

Beyond the 16-bit Card PCMCIA Standard

Today, a wide variety of 8- and 16-bit PCMCIA cards ("PC Cards") are available for adding various functions to a computing platform. Some of the more common cards include modems, LAN adapters (e.g. Ethernet), Flash memory cards, fast SRAM-based "silicon disk" cards, wireless communication adapters, and removable hard-disk cards. The speed of these cards is limited by the low bandwidth of their PCMCIA interface.

PC-Card applications which require substantially greater interface band-width are now beginning to appear. For example, a 100 Mbit/sec Ethernet adapter cannot realistically be implemented on a PC Card without resorting to the data-transfer speed of a CardBus interface. similarly, portable PC-based teleconferencing systems require a CardBus interface to the host platform.

Beyond raw performance, master-capable CardBus Cards make it possible to distribute processing intelligence throughout a system, offloading tasks from a single central processor. Such cards can be used to implement intelligent I/O adapters; for example, bus-mastering cards can execute their own data transfers. This capability is especially useful in platforms which contain hierarchical PCI buses, whose bus-isolation capability makes it is possible to effect parallel data transfers. In a system with this architecture, the system CPU may be processing out of RAM on the primary PCI bus, while data transfers are occurring between an I/O adapter and a buffer memory located on a higher-level PCI bus. In the extreme, it is possible to build a complete CPU card which is contained within a single CardBus Card.

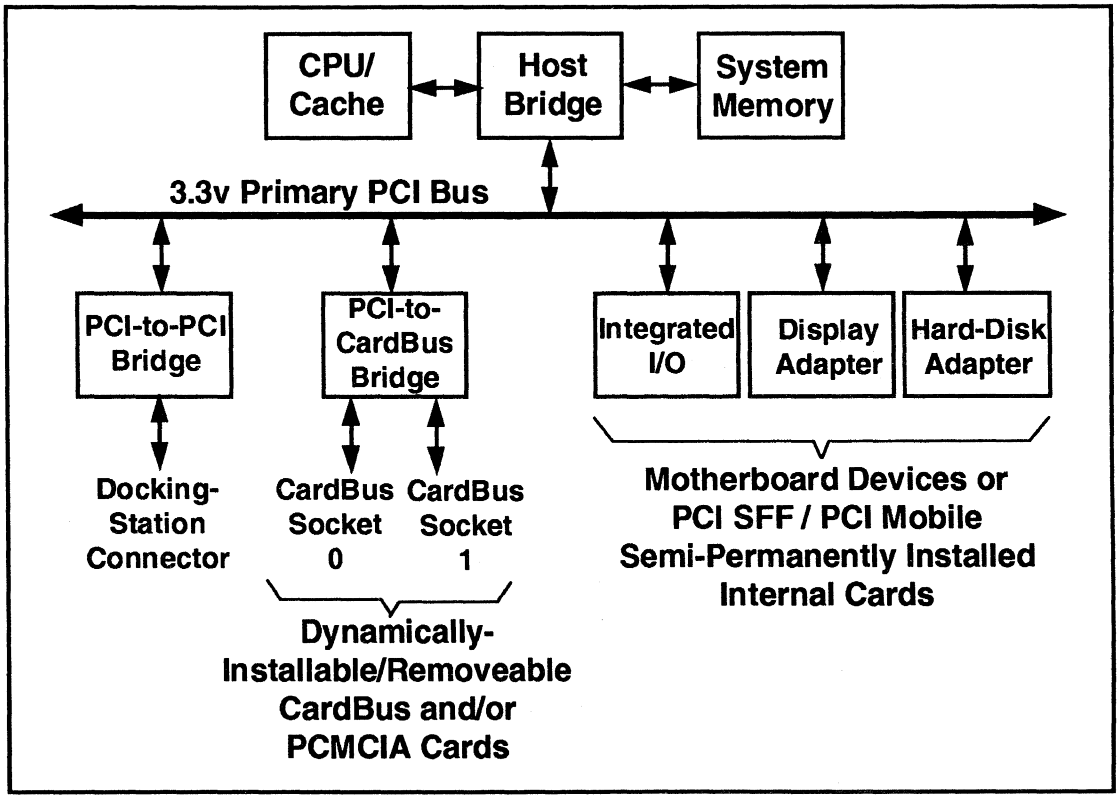

CPU/ Cache
Host Bridge
System Memory
3.3v Primary PCI Bus
PCI-to-PCI Bridge
PCI-to-CardBus Bridge
Integrated I/O
Display Adapter
Hard-Disk Adapter
Docking-Station Connector
CardBus Socket 0
CardBus Socket 1
Motherboard Devices or PCI SFF / PCI Mobile Semi-Permanently Installed Internal Cards
Dynamically-Installable/Removeable CardBus and/or PCMCIA Cards

Figure 2: CardBus-Capable PCI Portable Computer

Some Architectural Possibilities

In near-term portable computers, CardBus cards and PCI SFF or PCI Mobile cards may be connected to a 3.3v PCI system bus. In such a configuration, base motherboard functions can be augmented with internally-installed PCI SFF or PCI Mobile cards providing continually-needed functions, such as a LAN adapter. Shareable functions can be connected to the system via one or more CardBus slots which are overseen by a PCI-to-CardBus controller. The resulting portable system is schematized in Figure 2.

CardBus (and PCMCIA) Cards can contain significant portions of a computer's overall functionality, such a disk subsystem or a display adapter or large increments of memory. Both PCMCIA and CardBus include a "multi-function" capability through which multiple distinct functions can be integrated within one card, while retaining full software support from Card Services and Socket Services. (The main challenge in building such cards is in defining how to handle interrupts from multiple functions). This multi-function capability further broadens the scope of how system functions can be partitioned between motherboard and PC Card(s).

It is reasonable to consider a longer-term scenario in which major system functions migrate onto a set of CardBus cards. This might lead to a system which consists of a comparatively simple and inexpensive "mainframe" populated by one or more high-functionality CardBus Cards. The mainframe might consist of little more than a chassis, a power supply and a PCI bus populated with a handful of fairly generic functions, such as RAM.

Users could port their entire computing "environment" from one such mainframe to another, on a handful of CardBus cards. This would move the bulk of a system's expense onto readily-portable and customizable Cards, while allowing bulky and relatively inexpensive mainframes to be left in stationary locations. The processing-core Cards could equally easily be used in a compact battery-powered version of the mainframe for portable applications. All of this portability is based on a marriage of PCI's high performance with CardBus performance, hot-insertion and dynamic configuration capabilities.

Some Unresolved Issues

As a close relative of PCI, CardBus faces some of PCI's as-yet-unresolved issues. One of these involves how ISA-like "legacy" DMA can be performed using native PCI capabilities. DMA is presently used by certain types of I/O adapters, such as sound cards. One possibility is to provide one or more specialized data-transfer agents within a system, each of which uses burst-oriented PCI mastering and burst transfers to move data blocks. The difficult part of this solution is to also provide a hardware interface which "looks" like an Intel 8237-based ISA DMA subsystem to ISA-legacy software.

Another thorny problem involves how to provide an ISA-legacy interrupt facility without replicating all of the side-band signals which are used for this purpose in extant ISA systems. One solution might be to serialize (i.e. time-multiplex) multiple non-shareable ISA interrupts over an existing PCI/CardBus signal, using allowed transitions. A demultiplexer can then present the usual ISA interrupts to an Intel 8259-based ISA interrupt controller.

The two preceding problems are relevant to CardBus in that, through the foreseeable future, both ISA-style DMA and interrupts must be conveyed across both PCI buses and CardBus interfaces for X86-based platforms (since those systems constitute a large portion of all PC-class compute platforms). Whatever solutions are found for PCI will probably also work for CardBus, and vice versa.

In a rather different dimension, CardBus is an emerging technology, and thus presents some market adoption issues. The CardBus standard is now stable, but PCI-to-CardBus controller ICs are just beginning to make their appearance. CardBus card vendors cannot (and would not wish to) field high-performance CardBus cards until they perceive a sufficient level of interest in CardBus, as manifested through adoption by major platform vendors. Thus, it is likely that the rate of CardBus adoption will be determined by the rate of PCI adoption in portable systems (which is already high), and by the rate of incorporation of CardBus into major computer vendors' product lines.

Recently, there has been a strong up-turn in CardBus interest from computer manufacturers. It appears that this is being driven by a growing public perception that most of PCMCIA's (and CardBus') early compatibility problems have now been resolved. This is lending further momentum to PCMCIA as an attractive system-configuration technology.

Since PCI-to-CardBus controllers are not likely to be much more costly than PCI-to-PCMCIA controllers, platform vendors may be seeing an incentive to "future-proof" their products at minimal risk, even though there are not yet CardBus cards to take advantage of this prospective CardBus interface. The presumption is that CardBus-capable platforms will initially be used with PCMCIA R2 Cards (using backward compatibility as mandated in the CardBus standard), while CardBus cards are developed.

SUMMARY AND CONCLUSIONS

Taken together, PCI and CardBus are ideally-matched technologies for building versatile high-performance portable computing systems. PCI has already gained wide-spread acceptance as the system bus of choice in such systems. CardBus, which is closely related to PCI, offers a uniquely flexible mechanism for dynamically configuring a PCI-based system to meet the needs of particular applications and users. The major computer vendors are likely to show relatively rapid adoption of CardBus in their portable products over the next year or two. In addition, desktop computers should also begin to incorporate CardBus interfaces as a means for sharing high-performance adapters with portable computers.

REFERENCES

Personal Computer Memory Card International Association (PCMCIA), 5/3/94. PCMCIA CardBus Specification Draft, Revision 0.9.

QUICKRING™ TECHNOLOGY, THE HIGH PERFORMANCE PCI INTERCONNECT AND BRIDGE

Webster (Rusty) Meier Jr.
Principal Applications Engineer
National Semiconductor
M/S E-200
2900 Semiconductor Dr.
P.O. Box 58090
Santa Clara, CA 95052-8090 USA

ABSTRACT

PCI has rapidly become accepted as the high performance local bus for next generation of PCs. QuickRing Technology (QRT) is a low-cost high performance (200 M-bytes/sec) point-to-point ring-based interconnect architecture that enables the PCI architecture to achieve new heights of functionality and performance.

QRT supports a total Ring Bandwidth (16 node ring) of up to 1.7 Giga-Bytes / Second. When connecting multiple PCI Buses with QuickRing Controllers, each PCI bus becomes a node on the Ring.

This paper will examine a PCI interface to QRT. It will also examine how the QRT interconnect between PCI buses enhances application areas where high performance, concurrent transactions between PCI buses, multiple system interconnects and direct peer-to-peer transfers are necessary.

INTRODUCTION

The PCI Platform is rapidly becoming the industry standard local bus. It provides high performance and a consistent platform for system interconnects. There are issues associated with certain applications where the PCI performance cannot be fully realized. A solution is needed to overcome these obstacles of high performance, concurrent transactions, multiple system interconnects and direct peer-to-peer transfers.

QRT enables a high degree of system parallelism in a ring based topology, provides a simple generic client interface, and can be easily expanded to connect thousands of nodes with current silicon.

The PCI environment is a natural fit for QRT. QRT connects chips, cards, or racks together and is microprocessor and bus independent. QRT can help utilize full PCI performance across a variety of concurrent applications.

QUICKRING CONTROLLER CHARACTERISTICS

The QuickRing Controller (QRC) is a high speed point-to-point interconnect. A node on the ring interfaces to the QRC through two ports, an Upstream Port and a Downstream Port (see Figure 1).

The QRC transfers symbols from a source node to a target node. The source and target are each represented by a 4-bit field in the head symbol (see Table 2).

In a single ring, the maximum number of nodes is sixteen because of a 4-bit addressing scheme (see Table 2). The 4 HOP fields that additionally occur in the QRC Head allow thousands of nodes to be connected in a multi-ring architecture implemented by bridging the rings together. It should be noted that two QuickRing controllers can cross-connect their client interface ports (potentially no glue logic) to form a bridge (see Figure 2).

The QRC also has TWO separate 35-bit client interfaces - the transmit and receive ports. Each client port has a 32-bit symbol plus a 3-bit type field. The type field designates the 32-bit symbol type (basically Head, Data, Frame, or Null, see Table 1). There is also a single handshake signal for each client port to control the rate at which symbols are enqueued (Tx Port, TxOK) or dequeued (Rx Port, RxSTALL).

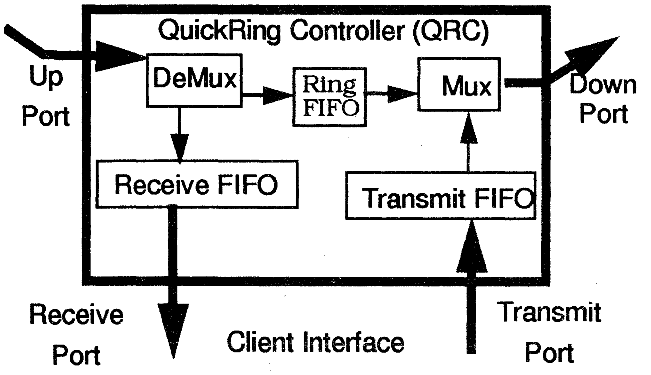


Figure 1: The QuickRing Controller (QRC)

The client transmit port can transmit an arbitrary length stream of symbols to a target. For example, 1 head followed by 1 million symbols could be fed into

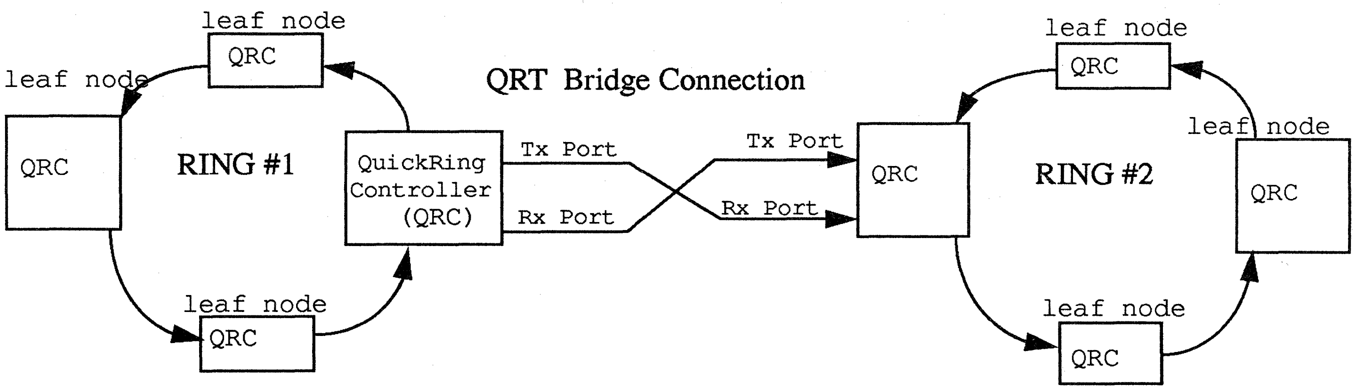


FIGURE 2: QRT Bridge in a Multi-Ring Topology

the transmit client port. The QRC will automatically packetize the stream into individual packets of one head followed by up to 20 symbols, then another head (same as previous) followed by 20 symbols ...etc., until the entire 1 million symbols has been transmitted (Packet description shown in Figure 3).

Table 1: Client Type Field Definitions (TxT/RxT)

Type	Name	Description
0	Directed Head	First symbol of a stream or packet, specifying the path to a single target.
1	Multicast Head	First symbol of a packet destined for one or more targets. This stream does not use the reservation based ring protocol.
2	Data	32-bit payload symbol is a Data.
3	Data-Tail	Data symbol which is the last symbol in a fixed length stream or packet.
4	Frame	Specially marked 32-bit payload symbol is a Frame.
5	Frame-Tail	Frame symbol which is the last symbol in a fixed length stream or packet.
6	Reserved	Future use
7	NoSymbol	No associated symbol

Table 2: Transmit & Receive Port Head Format

Rx/ TxT[2:0]	Rx/TxS[31:0]								
2:0	31:30	29:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
Type	Acc	Conn	Srce	Trgt	HOP				
0	0 Vary	Conn 0/1*	Srce	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT
0	1 Fixed	0	Srce	Trgt	Hop1	Hop2	Hop3	Hop4	HCNT
1	XX	0	Srce	Group Field		Multicast Field			

Conn = 1 indicated Low bandwidth
Type = 0 indicates a Directed Head to one Target
Type = 1 indicates a Multicast Head to multiple Targets

Currently there are four types of transmission available with the QRC (see Table 2):

- Low Bandwidth High Priority Packets, these packets have higher priority then the following types of packets.
- Directed Variable Sized Packets, the packet size is determined by the QRC, not by the system interface.
- Directed Fixed Sized Packets, the system interface determines the size of the packet (20 symbols or less).
- Multicast Packets, these packets may be sent to more then one node on the ring.

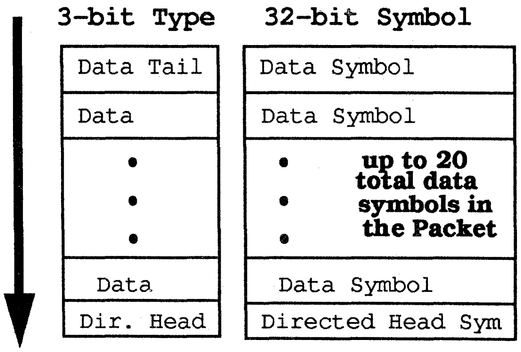


Figure 3: QuickRing Packet: 1 Head followed by up to 20 Data Symbols.

<u>QuickRing Technology (QRT) Reservation Based Protocol</u>

All QRC packets use a reservation based protocol (except multicast packets) to move across the ring to a target node. When the first symbol of a packet is enqueued a voucher is sent to the target node asking if the targets receive FIFO has space available to receive a packet (up to 20 symbols). If available, the target sends a ticket to the source node to launch the packet. Both the transmit and receive FIFO depth is greater then 100 symbols. As long as the receive FIFO has at least 20 symbols of vacency it will return a ticket to the source node (Figure 5).

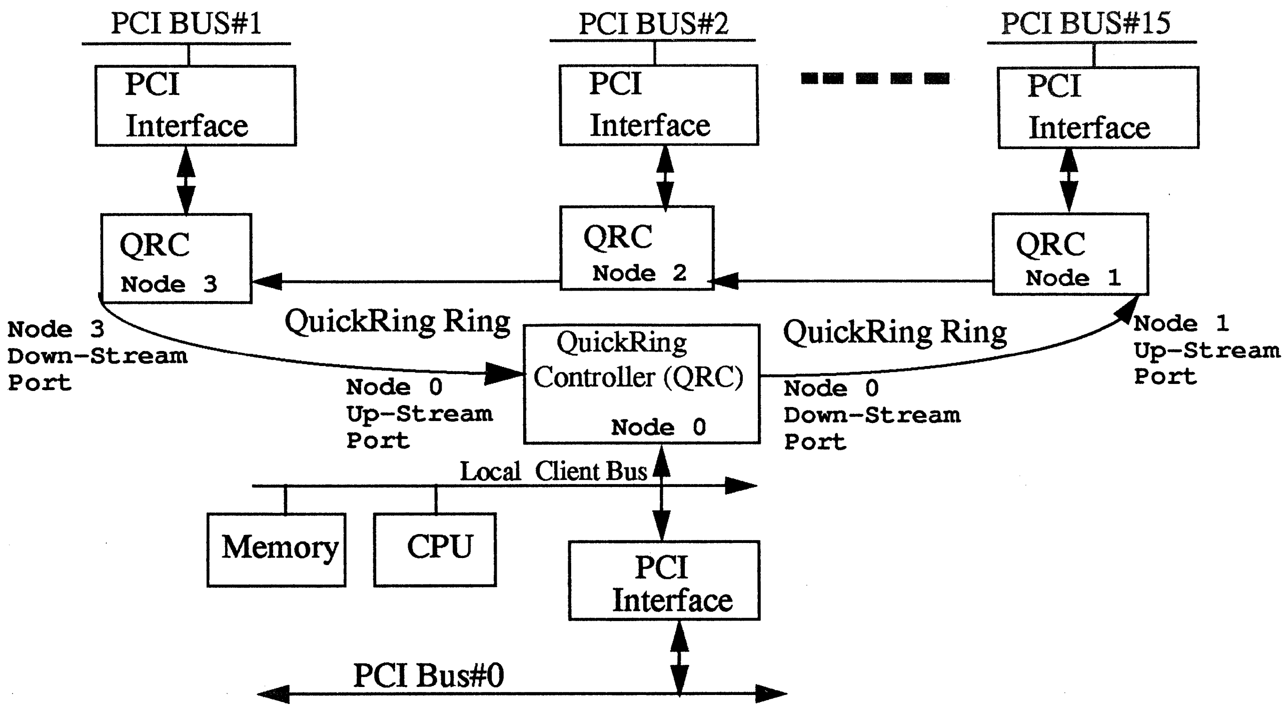


Figure 4: QRT applied to connecting PCI busses together

QRC Latency Issues

The QRC contains two data transfer protocols at the physical layer:

- Reservation based protocols (vouchers and tickets) for all packet types except multicast packets,
- Non-Reservation based protocol for multicast packets.

The following times relate to a reservation based protocol transaction on a two node ring, where the ring is initially idle (QRC Clock at 50MHz).

- First data enqueued into QRC transmit port (Node 0) to first data dequeued at target receive port (Node 1) = 40 clocks (800ns).

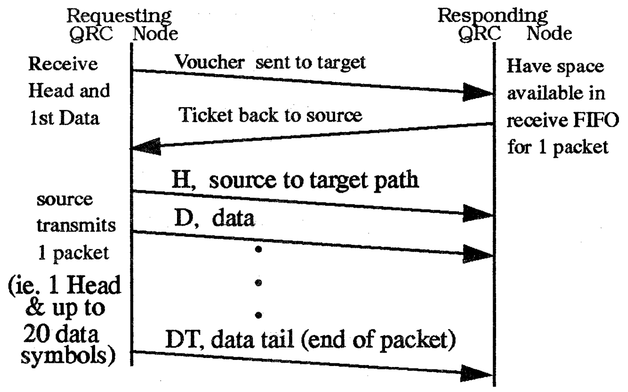

Requesting QRC Node
Responding QRC Node
Receive Head and 1st Data
Voucher sent to target
Have space available in receive FIFO for 1 packet
Ticket back to source
H, source to target path
D, data
source transmits 1 packet
(ie, 1 Head & up to 20 data symbols)
DT, data tail (end of packet)

Figure 5: Reservation Based Protocol

It is important to mention that it is extremely difficult for one or several QRC nodes to monopolize the Ring Bandwidth (BW). The reservation protocol tends to guarantee that no node/nodes will be starved in terms of the ring BW it may have access to, as long as the total desired BW of all nodes on the ring is less then the total available ring BW (200MBytes/sec).

Multicast packets (non-reservation based) will allow much faster transfers since they do not have the voucher/ticket overhead.

PCI / QRC APPLICATIONS AND BENIFITS

In any PCI application where high performance and concurrency between multiple boards is needed, The QRC is an appropriate solution (Figure 4). A few example applications are: Multimedia, Video Server, Network Hubs & Routers, Multiprocessing, Image Processing, Medical Imaging, DSP Systems, RAID array, Data Acquisition, PC Enhancement and PCI extender.

The following are some highlights of the QRC characteristics that a PCI application can take advantage of:

1. **High Performance and Concurrency:** A single node can transmit and receive data at a maximum rate of 200MB/sec (aggregate total per node of 400MB/Sec). Also multiple nodes on the ring may be transmitting and receiving simultaneously giving an aggregate ring bandwidth of up to 1.7 Giga-Bytes/sec.

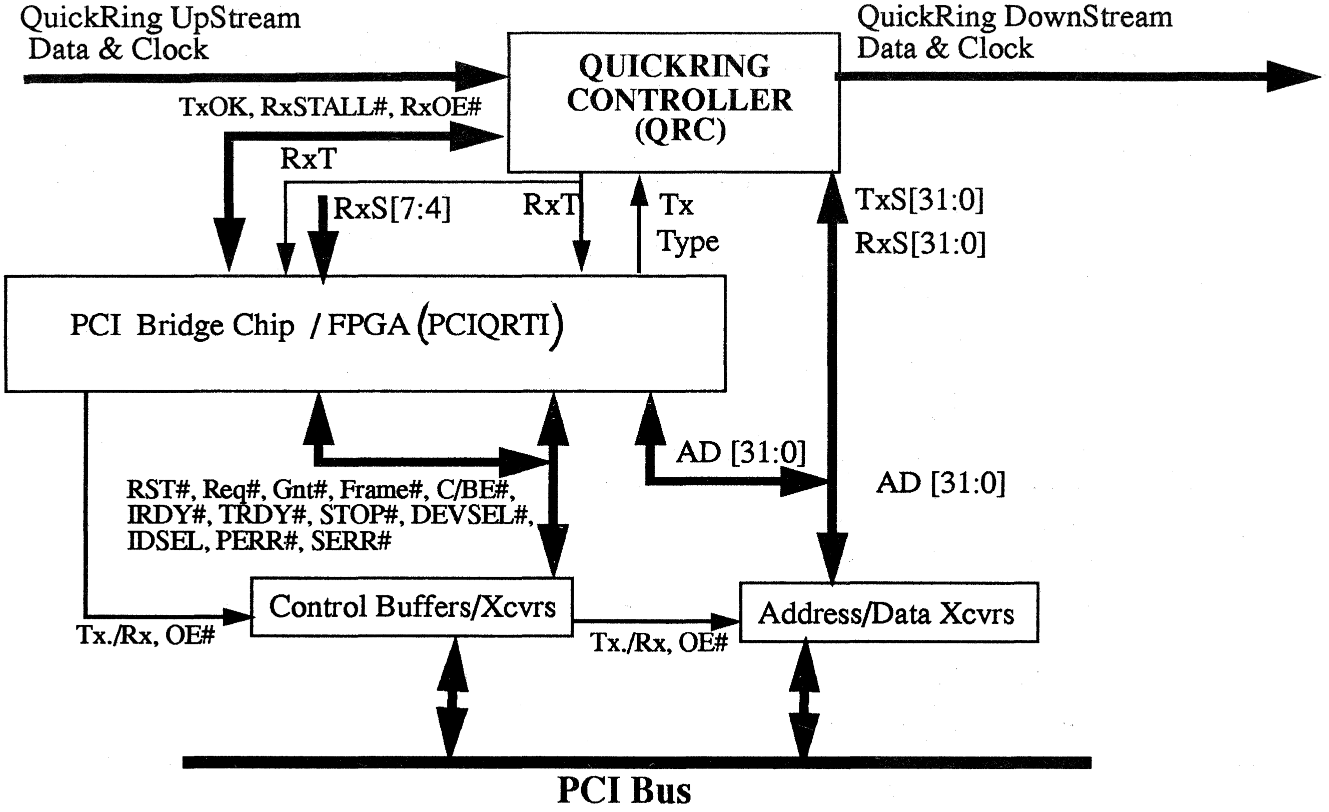


Figure 6: PCI QuickRing Technology Interface (PCIQRTI)

2. **Point to Point Low Voltage Differential Signaling (LVDS):** The QRC uses Low Voltage Differential Signalling (LVDS) which has high noise immunity and very low signal levels. The QRC can be run at full speed without crosstalk or noise problems between the LVDS and other PCI signals.
3. **Multiple System Interconnect Environment:** The QRC interconnect allows multiple card cage chassis, along with other equipment to be connected using an external cable connector or card to card connections using an internal card connector. The QRC can be used to interconnect boards, PCs, workstations, disk arrays, etc.
4. **Direct Peer-to-Peer transfers:** QRT overcomes the traditional system bus bottlenecks by providing an alternate communication route. QRT provides a direct link for peer-to-peer data transfers. For example, a video system could use QRT in the following manner:
   - a video camera could input video data and transfer it directly (via QRT) to a display card;
   - the display card could directly transfer the data (via QRT) to a compression card;
   - the compression card could then transfer the data (via QRT) to the storage card (i.e. disk array card)

THE PCI QRT INTERFACE (PCIQRTI)

A PCI to QRT Interface (PCIQRTI) is being designed. The initial version of this interface will be designed around the a PCI Bridge chip and an FPGA device (Figure 6).

The PCIQRTI features include:

- providing both a master and a slave interface to PCI.
- Compatibility with "PCI Local Bus specification Rev.2"
- Generating/Checking parity on PCI bus
- Allowing direct CPU transfers, DMA transfers, PCI write transaction bridge function, and control transactions.

The primary objective behind this design was to keep the PCIQRTI as simple as possible while allowing the system designer to take advantage of the high bandwidth, concurrency, and peer to peer transactions that the QRT offers.

The PCIQRTI Control Protocol Overview

The QRC is very efficient at moving large amounts of data between nodes on a Ring. The QRC inherently performs write transactions between nodes. In order to perform reads (or other non-write types of transactions) a protocol can be implemented on top of the inherent QRT protocol. A PCI QRT control protocol has been specified. In implementing this protocol the first symbol of data after the QR

head symbol is a **Frame Symbol**. The 32-bits of the frame symbol have been defined to allow implementation of a higher level protocol. Four of the bits are used to encode the particular PCI command (i.e. Interrupt, Memory/I/O/Configuration Read/Write, ..etc).

The Client Interface

To get a high level of performance out of the QRC it is necessary to include hardware capable of transmitting and receiving data every clock cycle (the receive and transmit clocks can run up to 50MHz). This could be provided by a transmit/receive DMA channel.

Only one simple receive DMA controller is needed because all Packets transferred across the interconnect could be Self-Sufficient fixed size Packets **(SSP)**, the packet size being one head followed by up to 20 symbols (Figure 7). The SSP includes:

- a **Head** that contains the source and target QRC node of the transfer (Head type = H),
- the **Frame Command,** marked as a frame type (F), necessary to perform the transfer (i.e. read, write,,, etc.),
- the **Address** of the transfer (address to read from or write to), marked as a data type (D),
- the **Data** (D) involved in the transfer. Up to 18 data symbols could be transferred per packet, the last data is a **Data Tail** type (DT).

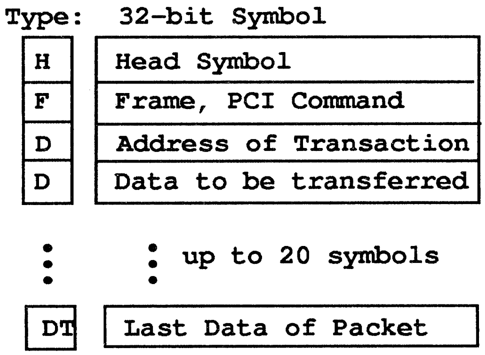


Figure 7: QRT Packet Format

This way the receive DMA would only need to concern itself with individual packets (it would have no concept of a stream). Each received packet would contain the information necessary to perform its transfer.

For example, one transmit DMA channel could be set up to transfer 160 Symbols across the QRC. The transmit DMA would divide the 160 symbols into 10 packets, each packet consisting of a head, command, address and 16 data symbols. The transmit DMA would automatically increment the address on each successive transferred packet and inset the particular frame command (read/write).

PCIQRTI, Types Of Transactions

There are four different types of data transfers available with the PCIQRTI:

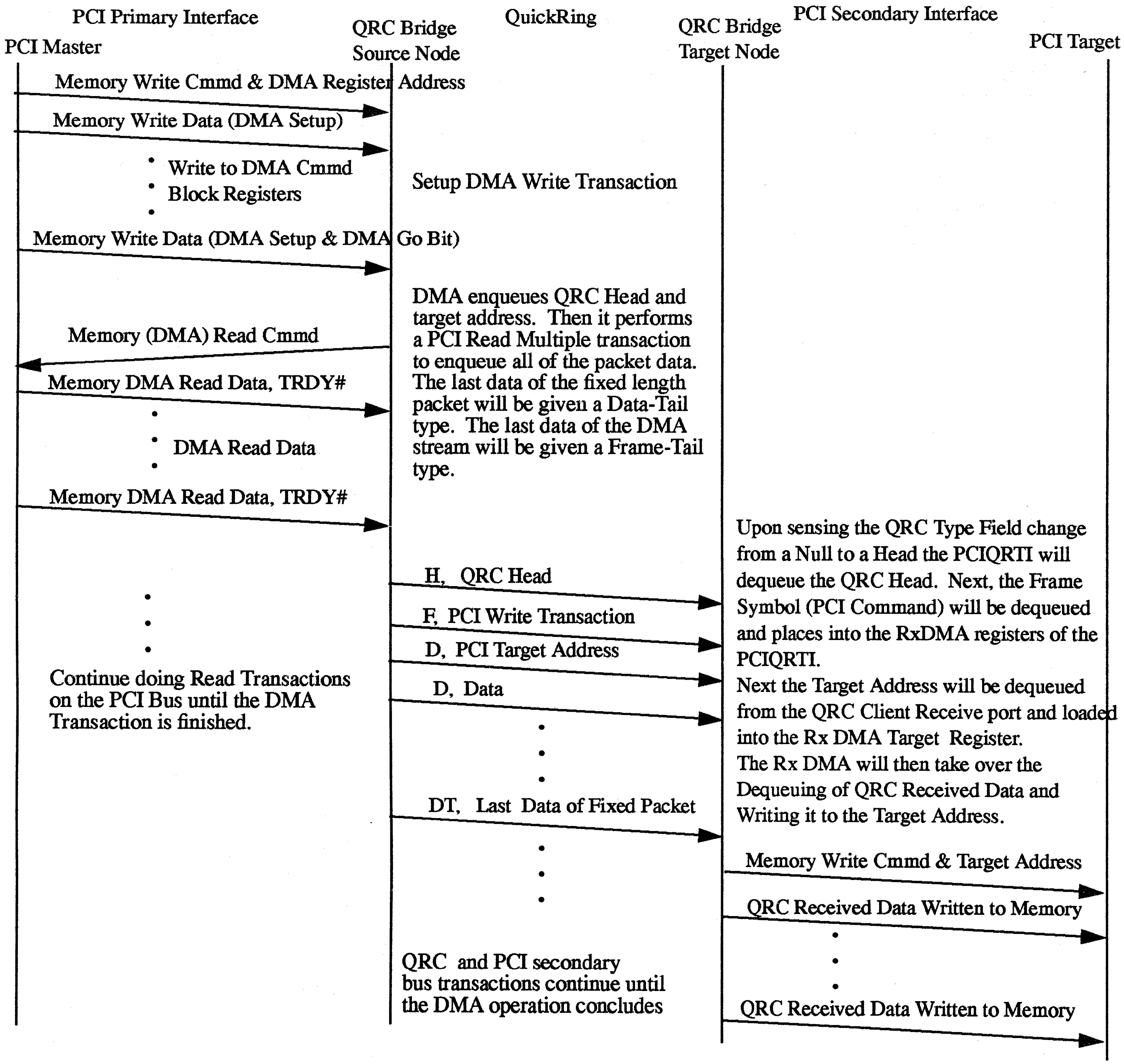
1) **DMA Self-Sufficient fixed size Packet (SSP) Write transfers**: In these transactions the source node CPU will write to the PCIQRTI DMA registers in order to set up the following write DMA (source and target memory addresses, last memory address of the source block, the QRC Head for the DMA, and the PCI control protocol frame symbol). Once the DMA has been set up and the "Go" bit has been set the PCIQRTI will start performing burst read accesses (up to 16 data transfers, 64 bytes, per PCI bus access). The transmit DMA will enqueue fixed size Self-Sufficient Packets (SSP) to the target node. Once the source DMA is finished the Host CPU may be interrupted to indicate this. Each of these SSP will be transferred across the QRC to the target node. The target node PCIQRTI DMA engine (seeing the frame symbol DMA bit [13] set) will dequeue each of these packets and perform PCI burst write transactions on the PCI bus. Upon detection of the "End of Transaction" symbol the receive DMA may interrupt its host CPU to indicate that the received DMA has terminated (Figure 8).
2) **Direct (SSP) Write Transfers**. The CPU will enqueue a SSP that look the same as the DMA packets described above, note that in this circumstance the DMA engine is only used in the target node.
3) **Control Packets** (CPU enqueued non-DMA). This allows the host CPU to enqueue fixed size packets of any protocol it desires. The target node will interrupt the host CPU to dequeue this packet.
4) **PCI Write Transaction Bridge Function**: This packet type is generated by setting the PCI write transaction bridge function (bit [0] of register 64h). If set this bit instructs the state machine to cause any PCI write memory transactions to the Base Address of the configuration space (address 14h) to get enqueued automatically into QRC. These transactions will automatically appear at the node specified by the DMA Head (register 58h). This bit set will cause the QRC to look like a PCI bridge for write memory transactions. The memory size for this area can be set in the configuration space base address register (10h).

The PCIQRTI is primarily a PCI interface to the QRC, although for PCI memory write transactions this interface can function as a PCI Bridge. In the initial boards the only PCI transactions that will be supported are:

- memory write, and,
- configuration read/write.

Future versions of this interface may be designed to support the complete set of PCI transactions.

Figure 8: Example of PCIQRTI DMA Block Write (Up to 64 Million 32-Bit Symbols


PCI Primary Interface
PCI Master
QRC Bridge Source Node
QuickRing
QRC Bridge Target Node
PCI Secondary Interface
PCI Target
Memory Write Cmmd & DMA Register Address
Memory Write Data (DMA Setup)
Write to DMA Cmmd Block Registers
Setup DMA Write Transaction
Memory Write Data (DMA Setup & DMA Go Bit)
Memory (DMA) Read Cmmd
Memory DMA Read Data, TRDY#
DMA Read Data
Memory DMA Read Data, TRDY#
H, QRC Head
F, PCI Write Transaction
D, PCI Target Address
D, Data
DT, Last Data of Fixed Packet
Memory Write Cmmd & Target Address
QRC Received Data Written to Memory
QRC Received Data Written to Memory

DMA enqueues QRC Head and target address. Then it performs a PCI Read Multiple transaction to enqueue all of the packet data. The last data of the fixed length packet will be given a Data-Tail type. The last data of the DMA stream will be given a Frame-Tail type.

Upon sensing the QRC Type Field change from a Null to a Head the PCIQRTI will dequeue the QRC Head. Next, the Frame Symbol (PCI Command) will be dequeued and places into the RxDMA registers of the PCIQRTI.
Next the Target Address will be dequeued from the QRC Client Receive port and loaded into the Rx DMA Target Register.
The Rx DMA will then take over the Dequeuing of QRC Received Data and Writing it to the Target Address.

Continue doing Read Transactions on the PCI Bus until the DMA Transaction is finished.

QRC and PCI secondary bus transactions continue until the DMA operation concludes

PCIQRTI Register Descriptions

There are two sets of registers contained within this interface:
- the PCI configuration space, and,
- the Transmit/Receive DMA registers.

CONCLUSION

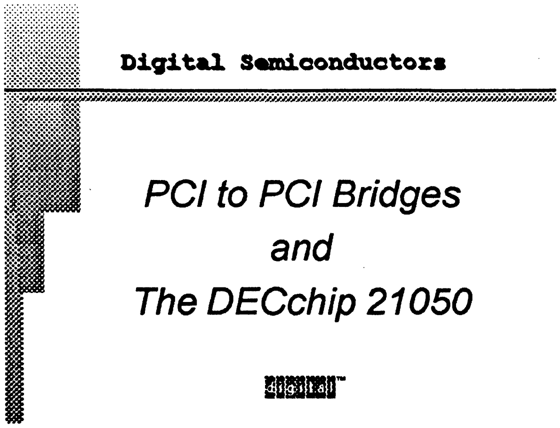
The QRC and PCI produce an ideal combination for applications that need low cost, high performance (up to 400MB/sec/node), low latency, concurrency, and conductibility between multiple PCI busses and systems.

REFERENCE LIST:

[1] QuickRing Bandwidth Calculations by Chakradher Reddy.

[2] National Semiconductor, June 1994, "The National QuickRing^{TM} Design Handbook".

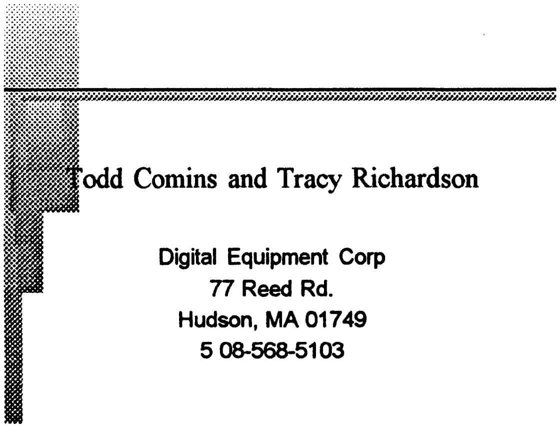
QuickRing^{TM} is a trademark of Apple Computer, Inc.



Digital Semiconductors

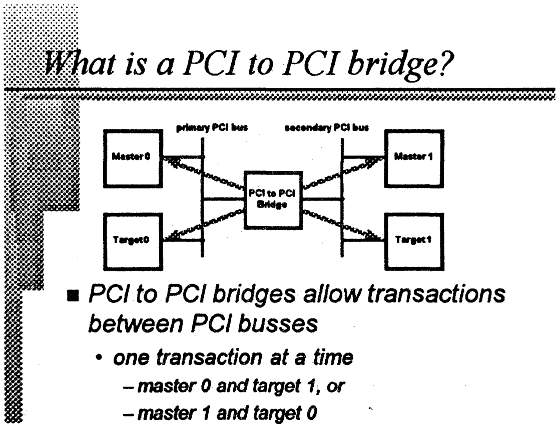
PCI to PCI Bridges
and
The DECchip 21050

digital™



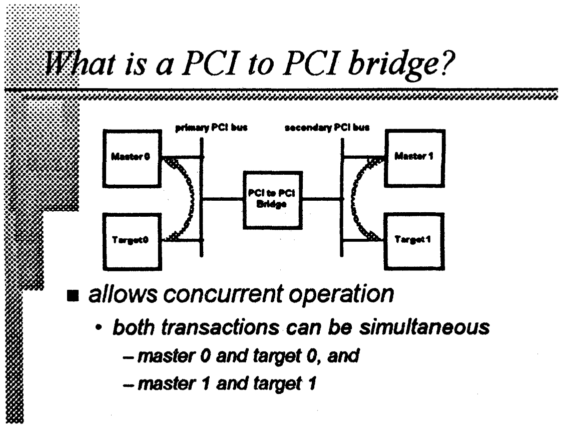
Todd Comins and Tracy Richardson

Digital Equipment Corp
77 Reed Rd.
Hudson, MA 01749
5 08-568-5103



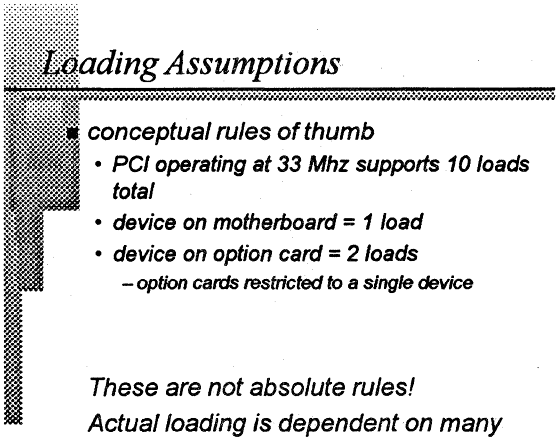
What is a PCI to PCI bridge?

- PCI to PCI bridges allow transactions between PCI busses
  - one transaction at a time
    - master 0 and target 1, or
    - master 1 and target 0



What is a PCI to PCI bridge?

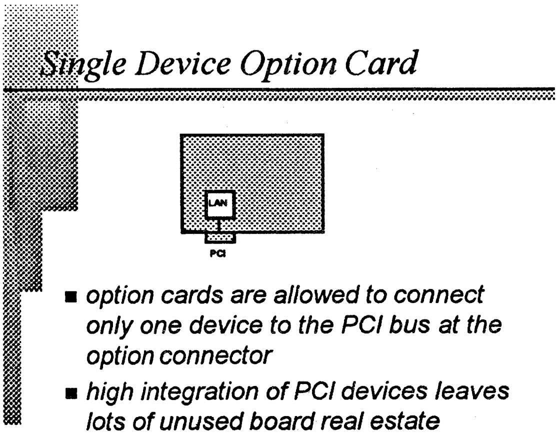
- allows concurrent operation
  - both transactions can be simultaneous
    - master 0 and target 0, and
    - master 1 and target 1



Loading Assumptions

- conceptual rules of thumb
  - PCI operating at 33 Mhz supports 10 loads total
  - device on motherboard = 1 load
  - device on option card = 2 loads
    - option cards restricted to a single device

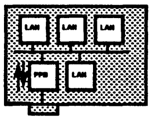
These are not absolute rules!
Actual loading is dependent on many



Single Device Option Card

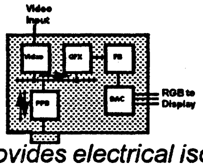
- option cards are allowed to connect only one device to the PCI bus at the option connector
- high integration of PCI devices leaves lots of unused board real estate

Multiple Device Option Card



- the PCI to PCI bridge (PPB) isolates the loading of the devices on the option card
  - enables multiple device option cards
- PCI transactions flow across bridge

Multifunction Option Card



- PPB provides electrical isolation
- PPB can isolate DMA traffic
  - bridge must support concurrent bus operation
  - Video to Graphics DMA does not cross the bridge

Transparency of Operation

- Once configured, a PCI to PCI Bridge is transparent!
  - a PCI to PCI Bridge does not have a device driver
  - no changes to devices or their drivers when attached behind a PCI to PCI Bridge

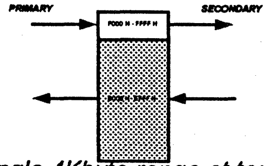
I/O and Memory Transaction Forwarding

- forwarding is controlled by address ranges
  - different registers for I/O, memory mapped I/O and prefetchable memory
  - base and limit registers define primary to secondary forwarding
    - secondary to primary forwarding is inverse
- PCI to PCI bridges provide three address ranges
  - I/O

I/O Transactions

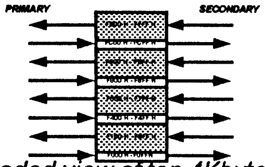
- I/O base and limit registers
  - 4 Kbyte granularity
  - ISA aware mode
    - blocks forwarding of ISA alias addresses
    - addressing is similar to EISA slot specific addressing
  - VGA aware and VGA palette snooping modes
    - only example of support for legacy devices
    - required to enable multimedia applications
- no write posting

I/O Addressing Example ISA Mode Disabled



- a single 4Kbyte range at top of I/O address space
  - inverse decoding is used for forwarding from secondary to primary
  - ISA aware mode is disabled
  - assuming 64 Kbyte total I/O address space

I/O Addressing Example ISA Mode Enabled

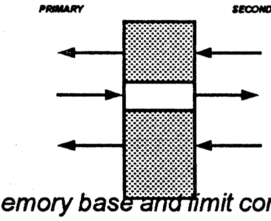


- exploded view of top 4Kbyte range from previous slide with ISA mode enabled (equivalent of EISA slot 15)
  - accesses are forwarded primary to secondary when in the bottom 256 bytes of every 1 Kbyte block in enabled range

Memory Mapped I/O Transactions

- memory base and limit registers
  - 1 Mbyte granularity
- writes can be posted
- read prefetching
  - memory read
    - no prefetching
    - reads may have side effects for memory mapped I/O devices
  - memory read line
    - implicitly specifies the prefetch of up to one cache line

Memory Addressing



- memory base and limit configuration registers
  - 1 Mbyte granularity
  - defines address range in which memory accesses are forwarded from primary to

Prefetchable Memory Transactions

- prefetchable memory base and limit registers
  - 1 Mbyte granularity
- writes can be posted
- all reads can be prefetched
  - optimization for bus masters that do not use memory read line and memory read multiple commands
  - devices identify prefetchability of memory address ranges via the standard base

Digital Semiconductors

The DECchip 21050

digital™

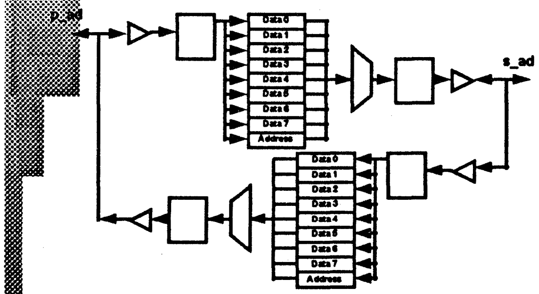
Digital is the Leader in PCI to PCI Bridges

- Digital is the recognized industry leader in PCI to PCI Bridges
  - Digital has the most technical expertise in the industry
  - Digital was the first to develop PCI to PCI Bridge products
- Digital Proposed and Chaired the PCI to PCI Bridge Workgroup
  - Lead the definition of PCI to PCI Bridge

DECchip 21050 Features

- Supports concurrent primary and secondary bus operation
- Provides 32 bytes of primary to secondary buffering
  - **posted write data**
  - **prefetched read data**
- Provides 32 bytes of secondary to primary buffering
  - **posted write data**
  - **prefetched read data**

21050 Data Path



DECchip 21050 Features

- supports I/O address range
  - 4 Kbyte granularity
  - ISA mode
  - VGA and palette snooping modes
- supports Memory Mapped I/O address range
  - 1 Mbyte granularity
- supports Prefetchable Memory address range
  - **1 Mbyte granularity**

DECchip 21050 Features

- Integrates secondary bus arbiter
  - six request/grant pairs
- Generates buffered clocks for secondary bus
  - six devices (in addition to the bridge)
- Operates at full 33 Mhz PCI clock frequency
- PCI lock transactions supported

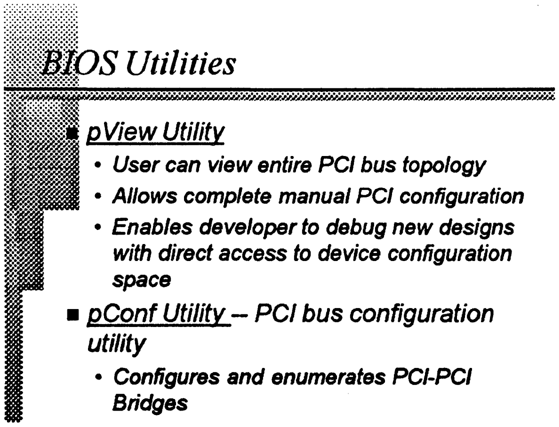
Is fully compliant with the PCI spec 2.0

Providing BIOS Support for the Bridge

- The DECchip 21050 does not need a device driver
- BIOS or initialization code must be able to recognize and configure a bridge
  - PCI to PCI Bridge support will be required to pass PCI SIG compliance testing
- We are working with the major BIOS vendors, system vendors, Microsoft and Intel to insure BIOS availability
- BIOS vendors now provide bridge

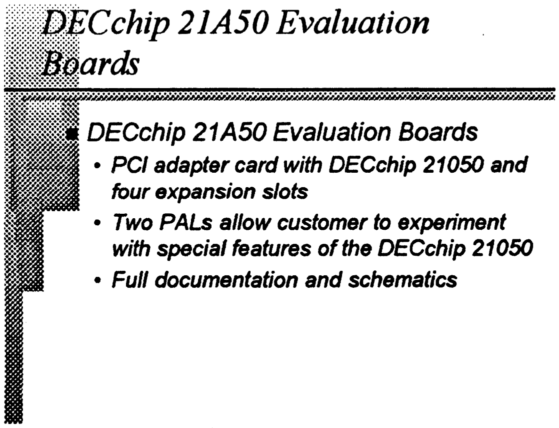
BIOS Utilities

- Temporary workaround for customers who do not yet have BIOS with PCI to PCI Bridge support
  - allows customers to configure systems and add-in cards with PCI to PCI Bridges
  - complete custom configuration possible
  - provides example code for PCI to PCI bridge enumeration and configuration
  - provides examples on PCI device mapping
- Currently on Digital's BBS – can be



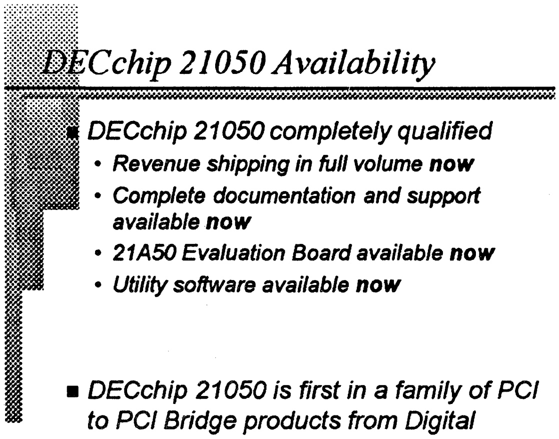
BIOS Utilities

- pView Utility
  - User can view entire PCI bus topology
  - Allows complete manual PCI configuration
  - Enables developer to debug new designs with direct access to device configuration space
- pConf Utility -- PCI bus configuration utility
  - Configures and enumerates PCI-PCI Bridges



DECchip 21A50 Evaluation Boards

- DECchip 21A50 Evaluation Boards
  - PCI adapter card with DECchip 21050 and four expansion slots
  - Two PALs allow customer to experiment with special features of the DECchip 21050
  - Full documentation and schematics



DECchip 21050 Availability

- DECchip 21050 completely qualified
  - Revenue shipping in full volume **now**
  - Complete documentation and support available **now**
  - 21A50 Evaluation Board available **now**
  - Utility software available **now**
- DECchip 21050 is first in a family of PCI to PCI Bridge products from Digital

EXPANSION ROMS FOR PCI DEVICES

Richard Holmberg
American Megatrends, Inc.
6145-F Northbelt Pkwy.
Norcross, GA 30071 USA

Abstract

The PCI bus standard opens up the PC architecture to new levels of performance and ease of configuration. This speed and flexibility also forces a new level of "responsibility" among expansion card hardware and software. PCI expansion ROMs are a critical component in the overall configurability of a PCI system. PCI expansion ROMs have added capabilities over the current ISA expansion ROMs such as a software programmable memory address and the ability to support more than one processor architecture.

Introduction

Personal computer expansion devices that participate in the boot process often require an expansion ROM which provides support for the device while loading the O/S. Examples of devices that often make use of an expansion ROM are VGA cards, SCSI host adapters, and network interface cards that load the O/S from a server. The expansion ROM mechanism used by PCI devices is much more robust that the traditional ISA or EISA expansion ROM standard. The major differences are listed below:

- PCI expansion ROMs may contain any number of code images. Each code image supports a different processor architecture.
- The base address of PCI expansion ROMs is software configurable, freeing the end user from having to manually set the ROM address of peripherals.
- PCI expansion ROM code is never executed in place. It is copied into system shadow memory before it is given control.
- When PCI expansion ROM code is initializing, it is running in read/write shadow.

Expansion ROM Contents

A PCI expansion ROM contains one or more code images. Each image contains code that is native to a different processor architecture. At this time only x86 and OpenBoot expansion ROM code types are defined. The system BIOS selects the appropriate code image for the system and copies only that one image into shadow RAM for execution.

Each image in a PCI expansion ROM contains an expansion ROM header and a PCI data structure. The format of the generic PCI expansion ROM header is shown below:

Generic PCI Expansion ROM Header

Offset	*Size*	*Description*
0h	byte	55h ROM signature
1h	byte	AAh ROM signature
2h	16h	Processor architecture dependent data
18h	word	Offset of PCI data structure within this code image.

For the x86 processor architecture, the data area at offset 2 contains the traditional information found in current ISA and EISA expansion ROMs. The format of the x86 expansion ROM header is shown below:

x86 PCI Expansion ROM Header

Offset	*Size*	*Description*
0h	byte	55h ROM signature
1h	byte	AAh ROM signature
2h	byte	Size of this code image in units of 512 bytes
3h	4	Entry point of code to initialize expansion ROM, accessed via a FAR CALL to this location.
7h	11h	Vendor defined data area
18h	word	Offset of PCI data structure within this code image.

The PCI expansion ROM header in each code image contains a pointer to the PCI data structure for that code image. The PCI data structure contains various information identifying the vendor and device, as well as a one byte code indicating the processor architecture supported by this code image. The format of the PCI data structure is shown below:

PCI Data Structure

Offset	*Size*	*Description*
0h	4	"PCIR" Ascii signature (P at offset 0)
4h	word	Vendor ID, same value as offset 0 in this device's configuration space
6h	word	Device ID, same value as offset 2 in this device's configuration space
8h	word	Offset of Vital Product Data (VPD) within this code image. The VPD structure is currently undefined.
Ah	word	PCI Data Structure size, set to 18h
Ch	byte	PCI Data Structure revision, set to 0
Dh	3	Class code, same value as offset 9 in this device's configuration space
10h	word	Size of this code image in units of 512 bytes
12h	word	Revision level of code image (vendor specific)
14h	byte	Code type of this image: 0 = Intel x86 architecture 1 = OpenBoot standard 2-FF = Reserved
15h	byte	Indicator Flags: Bit 7: If set, this is the last image in the expansion ROM, if clear, more image(s) follow. Bit 6-0: Reserved
16h	word	Reserved

Figure 1 shows an example expansion ROM for a PCI SCSI host adapter. The physical ROM is 64k in size and contains two code images. The first image is an x86 code image that is 16k in size. The second image is an OpenBoot code image that is 32k in size. There is 16k of unused space at the end of the expansion ROM. On an x86 architecture PCI platform, the system BIOS would search the expansion ROM, find the x86 code image, and copy that 16k image into read/write shadow memory before giving control to its initialization entry point.

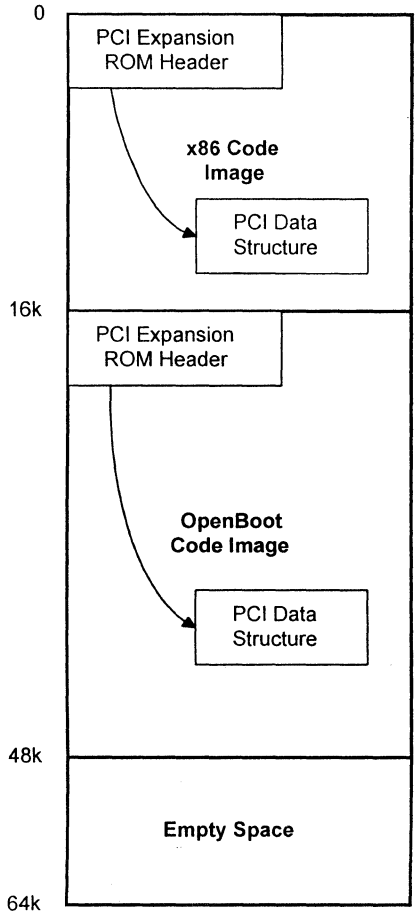


Figure 1: PCI Expansion ROM Example

Expansion ROM Base Address Register

PCI devices that need expansion ROMs must provide a mechanism that allows software to set the base address of the ROM. System software should be able to configure the PCI device so that its ROM is visible at any address in the 4GB memory space that is a multiple of the ROM's size. The PCI device's expansion ROM base address register must be used for this purpose.

There are a handful of PCI cards on the market today that use an expansion ROM but do not implement the expansion ROM base address register. Because these cards hardwire their expansion ROMs to a fixed address, they do not comply with the PCI specification and may not work on all PCI systems. Even PCI VGA cards whose expansion ROM code is almost always copied to and run from address C0000h, should not hardwire their expansion ROMs to this address. One of the goals of PCI is to provide peripherals that are fully configurable. Fixed address expansion ROMs clearly impede this objective.

The expansion ROM base address register is a 32-bit register located at offset 30h in a PCI device's configuration space. The format of this register is shown in figure 2. Bits 1 through 10 are reserved and should read as zeros. Bits 11 through n-1 should be hardwired to read as zeros. Bits n through 32 should be read/write. The value of n should be chosen so that 2^n is equal the size of the entire expansion ROM (including all of the images that it contains).

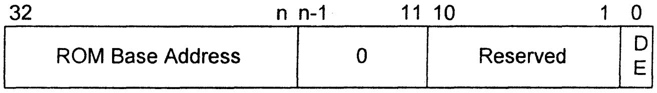


Figure 2: Expansion ROM Base Address Register

In order to determine the size of the memory block that will be occupied by the entire expansion ROM, the system BIOS typically writes FFFFFFFF to the expansion ROM base address register, then reads back the register, clears bit 0, inverts all of the bits, and finally increments this value. The BIOS uses the resulting size to find an unused block in the system's memory space. The base address of this block, which is a multiple of the size, is then written into the register.

Bit 0 of the expansion ROM base address register is used as a "decode enable." The PCI device responds to accesses to its expansion ROM only when bit 0 of the Expansion ROM base address register is set. The PCI device is not required to respond to any other memory or I/O address while the decode enable bit is set. During normal operation this bit is always clear. This allows a PCI device to share its internal address decoding logic between one of its standard base address registers and its expansion ROM base address register.

Expansion ROM Init Sequence

After the system BIOS has completed POST and configured all PCI devices in the system, it will initialize the PCI expansion ROMs. At this point each PCI device that implements an expansion ROM has been configured so that its expansion ROM base address register contains a base address which is typically near the top of the system's 4GB address space. Also at this point each PCI device's command register will have its I/O and memory decode bits enabled.

When initializing a PCI device's expansion ROM, the system BIOS first reads the expansion ROM base address register and writes it back with bit 0 set. This causes the PCI device to respond to reads in its assigned expansion ROM memory block. The expansion ROM should then be visible to the system.

The system BIOS (assuming an x86 system) then examines the first code image in the ROM. The system BIOS verifies that the code image contains a valid 55h AAh signature, and that the image checksums to zero. The BIOS will then find the code image's PCI data structure and examine its code type field. If the code type is x86 code (type 0), then the BIOS has found the proper code image. If the code type is not x86, then the system BIOS checks bit 7 of the indicator flags field for the presence of another code image in the expansion ROM. The system BIOS repeats this search until an x86 image is found or until all code images in the ROM have been checked.

Upon finding a valid x86 code image in the expansion ROM, the system BIOS will use the size field in the image's header to find a region of shadow memory large enough to hold the x86 code image. This region will start on a 2k boundary in the range C0000 through DF800. The BIOS makes this shadow RAM region readable and writable, and then copies the x86 code image from the expansion ROM into the shadow RAM region. The BIOS then disables the PCI device's expansion ROM by reading the expansion ROM base address register, clearing bit 0, and then writing it back.

The next step in the initialization sequence is to give control to the expansion ROM's init routine. This is done by making a far call to offset 3 of the x86 code image in shadow RAM. When making this call, the system BIOS passes some valuable information to the ROM's init routine:

Inputs to Expansion ROM Initialization Routine

Register	*Description*
AH	PCI Bus number of device that supplied the expansion ROM
AL	Bits 7-3: Device number of device that supplied the expansion ROM Bits 2-0: Function number of device that supplied the expansion ROM

The expansion ROM code then initializes its associated PCI device. As part of its initialization a PCI device's expansion ROM will usually need to determine the configuration of the device that it supports. There are several BIOS services that may be used by the expansion ROM code to accomplish this. These services are described in the next section.

When initializing the PCI expansion ROM code will be running in read / write shadow memory. PCI expansion ROMs can make use of this feature by building tables and data structures directly in their code segments. Some ISA and EISA expansion ROMs have traditionally reserved a block of the system's base memory for this purpose. Building data structures directly in the ROM's code segment is a much cleaner approach and should be used whenever possible. It should be noted that the system BIOS will write protect all memory containing expansion ROM code before the system boots. So any data structures contained within the ROM's code segment cannot be modified after the ROM's init routine exits.

The PCI specification outlines a mechanism that allows expansion ROMs to reduce their size after initialization. This feature allows PCI expansion ROMs to make efficient use of the precious 128k of memory available to them.

To make use of this feature expansion ROMs should position their initialization code at the end of the code image. Code and data needed once the system boots should be located at the start of the code image. Before returning from its initialization routine the expansion ROM code can change its size by adjusting the size byte at offset 2. After this is done the expansion ROM code must also adjust a byte anywhere in the new smaller code image so that the new code image will checksum to zero.

It is also possible for an expansion init routine to effectively remove the entire code image from memory. This is done by setting the byte at offset 2 to zero to indicate a new size of zero bytes. This feature may be useful if a PCI device decides during its initialization that it will not participate in the boot process. One example of this situation is a SCSI host adapter that does not find any bootable drives attached.

When finished initializing, the expansion ROM code should execute a RETF instruction to return control to the system BIOS.

Determining PCI Device Configuration

The PCI configuration mechanism is dynamic, a device's configuration may change from one boot to the next. Any software that communicates directly with a PCI device must deal with this dynamic configuration. The system BIOS assigns I/O addresses, memory addresses, and IRQ levels to PCI devices before giving control to the device's expansion ROM. In most cases PCI expansion ROM code will need to determine which I/O address(es), memory address(es), and IRQ level(s) have been assigned to its card each time the system boots.

The first task a PCI expansion ROM faces is locating its associated card in the system. To locate a PCI device three pieces of information are needed: the device's PCI bus number, its device number, and its function number. The system BIOS makes this task very easy; it passes this information to the expansion ROM code when control is transferred to the ROM's initialization routine. As described above, the AH and AL registers contain the bus, device, and function number of the PCI device that supplied the expansion ROM. This information should be saved by the expansion ROM code, and used in subsequent calls to read from the PCI device's configuration registers.

When trying to locate its PCI device, PCI expansion ROM code should not use the "Find PCI Device" BIOS service (Int 1A B102). If there are two devices in the system with the same vendor and device IDs, this BIOS function may return the location of either one. Use of this function by expansion ROMs makes it impossible to for the system BIOS to prioritize boot devices.

The system BIOS supplies a set of functions for reading from the configuration space of PCI devices. When using these functions the caller must supply the bus, device, and function number of the PCI device to read. These are the same values that were passed to the expansion ROM code in AH and AL (described above). PCI expansion ROM code should not use IN or OUT instructions to access PCI configuration space directly. The following table describes the functions available for reading and writing to PCI configuration space. The functions are called using an Int 1Ah with the AX register set to the indicated value.

PCI BIOS Configuration Space Functions

AX	*Description*
B108	Read configuration byte
B109	Read configuration word
B10A	Read configuration dword
B10B	Write configuration byte
B10C	Write configuration word
B10D	Write configuration dword

Summary

The PCI bus standard opens up the PC architecture to new levels of performance and ease of configuration. This speed and flexibility also forces a new level of "responsibility" among expansion card hardware and software. PCI expansion ROMs are a critical component in the overall configurability of a PCI system. Only when a PCI system's expansion ROMs are implemented correctly, can it meet the performance and usability objectives set forth in the PCI specification.

References

PCI Special Interest Group, April 30, 1993, PCI Local Bus Specification Revision 2.0

PCI Special Interest Group, August 26, 1994, PCI BIOS Specification Revision 2.1

Meeting the Challenges of PCI BIOS Development

Maxwell G. (Greg) Paley
Vice-President of Engineering Worldwide
Award Software International, Inc.
777 East Middlefield Rd.
Mountain View, CA 94043
(415) 968-4433/0274 (fax)

Award Software has been a major player in meeting the early challenges of PCI bus technology, and has worked closely with the PCI SIG in meeting those challenges. Award sees the increase in processor power offered by the latest generation offering of CPU manufacturers as creating a critical need for high-performance I/O subsystems in order to achieve a balanced system design.

PCI/Plug and Play
Support in PhoenixBIOS

PCI 2.1 Compliance

Frances Cohen
Principal Engineer
Phoenix Technologies, Inc.
2575 McCabe Way
Irvine, CA 92714
Ph. (714) 440-8323 Fax (714) 440-8300

Overview

- **New Plug and Play Requirements**
- **POST**
- **RunTime Services**
- **IRQ Routing**
- **Windows 95 Issues**
- **In Conclusion**

Plug and Play BIOS Requirements

- Responsible for PCI Card
  - Detection
  - Configuration
- Runtime Interface -- Provides a Way for Operating Systems to Configure Hardware on the Fly

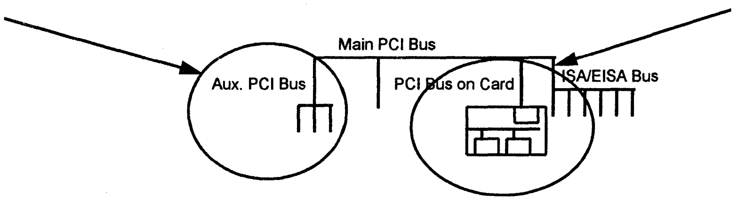
NEW

PCI - System BIOS POST

- Allocates Resources for Conflict-Free Operation
  - Avoids 10-Bit I/O ISA Aliases
  - Optimizes IRQs Dedicated to PCI
  - Shares IRQs Only When Necessary

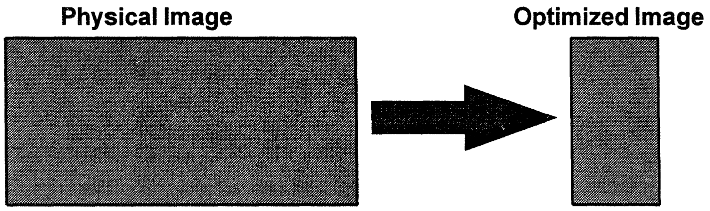
PCI - System BIOS POST

- **Writes Configuration Space Registers**
  - **Detects and Configures PCI-PCI Bridges**
  - **Writes Resource Allocations to Configuration Registers**
  - **Writes to Control Registers and Enables Devices**



PCI - System BIOS POST

- **Shadows and Initializes Expansion ROMs**
  - **ROMs Reduce Their Foot-Print After Initialization**
  - **Optimizes ROM Space by Shadowing Each ROM After Reduced Foot-Print**



PCI - Sys BIOS Runtime Services

- 16 and 32-Bit Interfaces
- Functions Supported
  - Find PCI Device/Class Code
  - Read/Write Configuration Space Registers
  - Get PCI Interrupt Routing Options (0eh)
  - SET PCI Hardware Interrupt (0fh)

PCI - Sys BIOS Runtime Services

- Future Operating Systems Will Use Runtime Interfaces!
- Issues with Direct Hardware Access
  - BIOS Can Cover-up Platform and Chipset Anomalies
  - Embedded Devices Return Wrong Class Codes
  - Configuration Access Mechanisms Need Work-Arounds

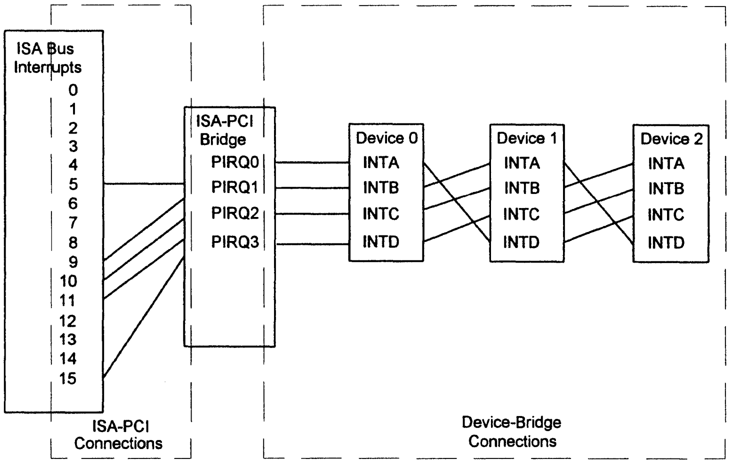
PCI - Sys BIOS Runtime Services

- Platform Slot Information Returned
  - The Masked Slot Number on the Board Can be Associated with the PCI Device
- Operating Systems Can Fully Understand and Control Interrupt Routing

PCI - IRQ Routing

- PCI Slot Interrupt Lines are Routed Through the Chip Set to the Programmable Interrupt Controller
- Platform Specific BIOS is Required to Control the Routing of IRQs on Host PCI Buses and Behind PCI-PCI Bridges

PCI - IRQ Routing Diagram

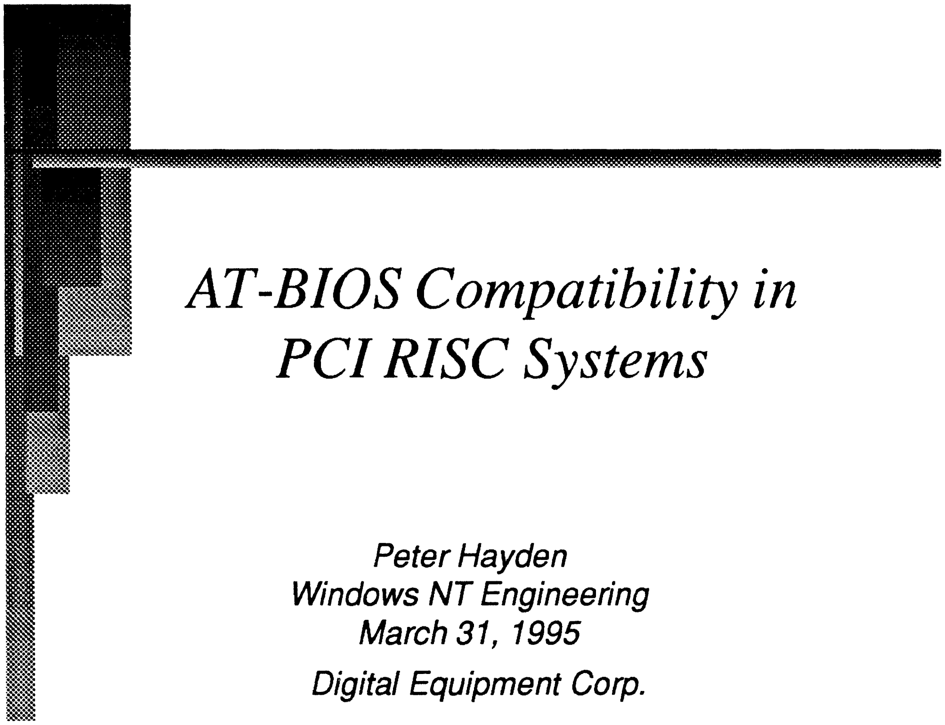


Windows 95 and PCI

- **Windows 95 Does Not Configure Cards Behind A Bridge**
- **Windows 95 Will Enable Cards that Were Disabled by the BIOS**
  - **IRQ register must already be loaded**
  - **IRQ routing must already be established**

In Conclusion...

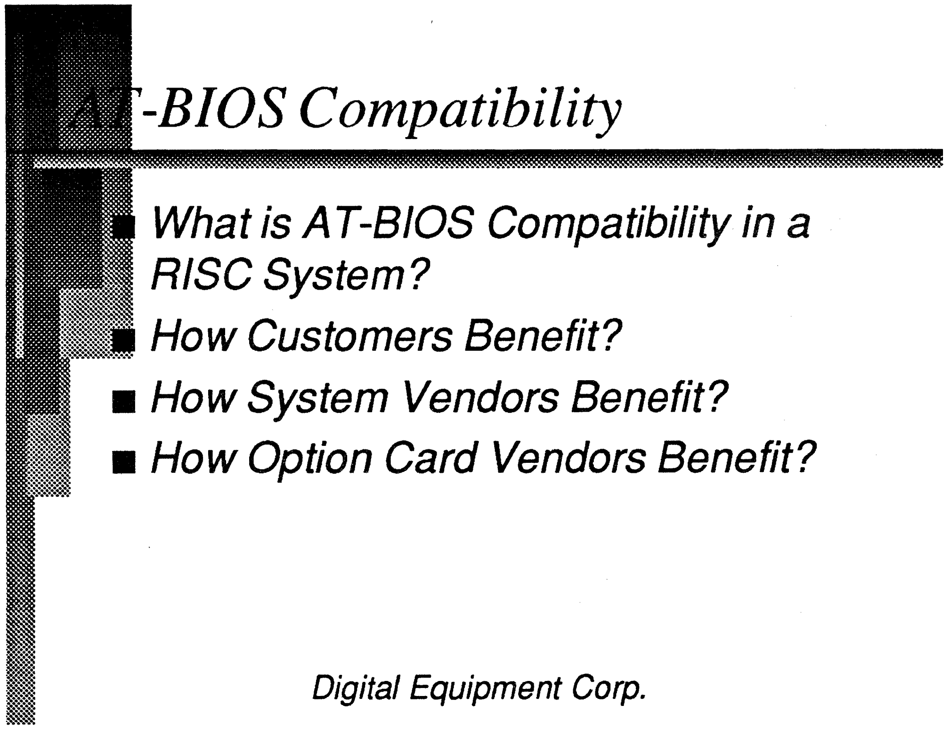
- BIOS has changed to meet the needs of Plug and Play PCs
  - POST now configures add-in hardware
  - Runtime interface allows all PCI devices to be controlled
  - Supporting Windows 95 adds new requirements
- PhoenixBIOS 4.0 incorporates complete Plug and Play Support



AT-BIOS Compatibility in PCI RISC Systems

Peter Hayden
Windows NT Engineering
March 31, 1995

Digital Equipment Corp.



AT-BIOS Compatibility

- What is AT-BIOS Compatibility in a RISC System?
- How Customers Benefit?
- How System Vendors Benefit?
- How Option Card Vendors Benefit?

Digital Equipment Corp.

What is AT-BIOS Compatibility?

- Non-x86 systems capable of using AT-BIOS extensions on option cards
- Video support via graphics board's video BIOS
- Disk I/O via SCSI BIOS
- System firmware emulates x86 CPU, PC hardware, PC system BIOS.

Digital Equipment Corp.

How Customers Benefit

- Expectations fulfilled
  - Options that plug in should just work
- Broader selection of options
- Lower costs by using commodity components

Digital Equipment Corp.

How Customers Benefit

- *Many RISC systems only support a short list of specific options*
  - *Expanding support requires update of system ROM.*
- *Other systems support OpenBoot*
  - *Only supported by a few PCI options*
  - *Doesn't support ISA, EISA, or VL options*

Digital Equipment Corp.

How System Vendors Benefit

- *Supports cards for ISA, EISA, and VL buses as well as PCI*
- *Take immediate advantage of new graphics developments*
- *Not restricted to boards supporting OpenBoot*
- *Eliminates cost of engineering and deploying OpenBoot support*

Digital Equipment Corp.

How System Vendors Benefit

- Eliminates need to upgrade system firmware to support new options.
- Broader option support enables more specialized configurations for vertical markets.
- Co-operative development with option vendors not required.

Digital Equipment Corp.

How PCI Card Vendors Benefit

- Sell into growing RISC market with your standard PC offerings.
- Co-operative development with system vendors not required
- Eliminates need to develop and deploy OpenBoot ROM
  - Lower engineering and material costs
  - Simpler distribution

Digital Equipment Corp.

Summary

- *AT-BIOS Compatibility provides platform independence like OpenBoot on PCI.*
- *Added advantage of supporting ISA, EISA, and VL cards.*
- *Customers, System Vendor, and Option vendors all benefit significantly.*
- *Available from Digital - Shipping today*

Digital Equipment Corp.

PCI Architecture BIOS Implications

Tim Hennessy
Product Development Manager
SystemSoft Corporation
313 Speen St.
Natick, MA 01760
(508) 651-0088/8188 (fax)

Since the PCI architecture has been introduced to the personal computer, there has been significant impact on system BIOS. The system BIOS has required considerable enhancement to support both PCI devices and a Plug and Play aware environment. The purpose of this discussion is to focus on the areas that impact the BIOS directly.

The primary BIOS consideration involves recognizing and automatically configuring PCI devices, both boot and non-boot devices. The configuration of these devices must be done with regard to other devices in the system, regardless of whether they are add-in devices or motherboard-resident devices, statically or dynamically configurable. The two specifications which govern the BIOS requirements are the PCI 2.1 BIOS specification and the PnP 1.0a BIOS specification. While these specifications address the primary BIOS issues, there are areas in which BIOS designers and, as a direct result, system designers, can benefit from greater enhancements and improvements to the BIOS requirements as outlined in these specifications. SystemSoft will reveal what some of these enhancements involve.

Of late, there has been a great deal of attention paid to PCI to PCI bridges. In being the first vendor to provide BIOS support for such bridges, SystemSoft has gained significant expertise in the area. The discussion will focus on the differences between peer to peer and hierarchical PCI bridge architectures as well as the BIOS implications and system recommendations for each.

Multimedia Applications and PCI

Mr. Mark S. Wodyka
Vice President
AITech International Corporation
47971 Fremont Blvd.
Fremont, CA 94538
Ph. (510) 226-8960 Fax (510) 226-8996
Internet E-Mail: Markw@Aitech.com

Multimedia, a sometimes ambiguous term in the computer industry, is generally accepted as the convergence of computers with video, film, sound, animation, photos and text. The various media are digitized so that a computer can understand and manipulate them, allowing the user to control the presentation and delivery of information.

Multimedia, the fastest growing computer technology arena, has gone form four billion dollars in 1990 to a projected twenty-five billion dollars by the end of 1995. The major application areas for multimedia are in desktop video, presentations, training, entertainment, kiosks, desktop video networking and telecommunications.

The discussion overviews the application areas of multimedia which will take advantage the rich features of the Peripheral Component Interface. Specific attention will be given to applications where previous limitation of CPU speed and bus transfer rates hampered the development of multimedia applications, such as: desktop video editing, full motion video capture, audio and video playback, and desktop video networking.

DSP-Accelerated Multimedia for PCI

Mark Clayton
Ariel Corporation
433 River Road
Highland Park, NJ 08904 USA
Phone: 908.249.2900
E-Mail: Mark.Clayton@ariel.com

Abstract

The advantages of using DSP for advanced audio and video processing are well-known to users of ISA computers. The PCI platform, along with recent advances in DSP technology enable even higher performance systems capable of real-time video processing, PC-based teleconferencing and acceleration of time-consuming data compression algorithms such as motion JPEG and MPEG.

The Peripheral Component Interface (PCI) bus is a high-speed local bus used to interconnect multiple 32-bit devices with a synchronous, processor-independent interface at rates up to 132 Mbytes/sec. PCI supports plug-in cards as well as motherboard devices, multimastering capabilities, auto configuration on power-up and full parity for mission-critical applications. PCI offers significant advantages over the ubiquitous ISA bus such as significantly faster data throughput, ease of installation and setup, reliable bus mastering capability and platform independence.

Texas Instruments' TMS320C80 'MVP' DSP chip is the latest in the TMS320 series of DSP chips from the company and represents a significant departure from previous generations. The C80 chip includes four integer DSP cores and a 32-bit floating-point RISC core which all work in parallel. The C80 performs up to two billion operations per second and is well-suited to real-time processing of video and/or audio signals.

Many multimedia applications can make use of the advantages of the C80 and PCI bus. For example, teleconferencing requires real-time audio and video decompression, both of which require massive amounts of processing power. For multimedia authoring, many existing and emerging compression/decompression algorithms are asymmetrical with respect to processing complexity. For example, MPEG video requires many times the processing power to compress a video frame as it does to decompress it. Also, video and audio post-processing for preparing multimedia program content can significantly burden an unaccelerated system.

PCI for Multimedia

The PCI bus has many advantages for multimedia authoring workstations and for high-performance end-user systems. The installed base of ISA-based MPC systems typically include a hardware-accelerated video card (possibly with a VL-Bus local bus interface) and an ISA-based sound card and modem card. Any real-time video (such as video-in-a-window) must be accommodated by special-purpose cards and/or supplemental over-the-top busses to handle the extremely high data rates. Data rates for audio input or output are more modest (176 Kbytes/sec for CD-quality stereo audio) so are easily handled via ISA plug-in cards. The data rates for MIDI and serial fax/modems are even lower so little performance can be gained by the speed advantage of PCI. This is not to say that there are other advantages for PCI implementations of these devices (such as automatic configuration and portability), but in these cases the ISA bus performs adequately from a data flow perspective.

$$\boxed{640\text{w} \times 480\text{h} \times 16\text{bpp} \times 30\text{fps} = 18.432\text{Mbytes / sec.}}$$

Uncompressed, full-frame live video requires a minimum of 18 Mbytes/sec, which is way beyond the capabilities of the ISA bus and can challenge even well-designed PCI systems. Even if the system handles the video data flawlessly, other bus activity (such as graphics drawing, disk or network I/O) may be impeded. A better solution for playback is to pass compressed video data over the PCI bus and add real-time decompression hardware to the video subsystem. This decompression hardware can include a dedicated function decompressor (such as C-Cube or Indeo) or a general-purpose processing element like the C80 for compression or decompression. And, unlike a chip dedicated to a particular function and algorithm, the C80 can be programmed for virtually any audio or video compression scheme.

Chip vendors have recognized the large potential market for multimedia playback systems (both desktop and set-top) and are already offering products that can perform these dedicated decompression algorithms many times faster than the host computer or conventional DSP chips. These chips offer substantial price/performance advantages for the high-volume applications, but are only useful for decompression of video and/or audio encoded in a way that the chip is built to understand. Obviously, future enhancements (or bug fixes) to the encoding algorithm cannot be incorporated into any existing silicon while a programmable solution can adapt to new and improved compression algorithms.

Another aspect to consider is asymmetry in the computational requirements of compression and decompression. Computational complexity of the compression algorithm is justifiably traded off in favor of both more effective compression and simplification of the decompression algorithm. This makes sense for authoring of multimedia material as the compression process occurs just once during the mastering of the media while decompression is performed every time each copy of the material is played. A dedicated DSP can ease the burden of the compression of the material, to the point where literally hours of computer time may be saved in some instances. Consider a situation where 45 minutes of 352 x 240 pixel video at 30 frames per second is to be compressed using the MPEG-1 compression standard. Using a fast PC, each frame can be compressed in a few seconds, but, since the PC can't compress the video as it comes in (i.e. in real time), the live uncompressed video must first be recorded to random-access storage or else an expensive frame-accurate video player can be used to present each frame in sequence. Assuming that the PC can compress one frame per second, 45 minutes of video would require 22.5 hours of CPU time during which the PC could do little else. A single TMS320C80 can compress this video in real time so the job would be completed in 45 minutes and there would be no need to store the raw video at all.

A C80 DSP implementation on PCI offers many advantages over ISA implementations. With a PCI interface to the host, high data rates on and off the DSP board can be achieved on a continuous basis. As demonstrated above, real-time processing of video and/or audio can make a huge difference in computation time as well as other required system resources. The full multimastering capabilities of PCI can be used again to greatly accelerate the data processing job at hand. For example, a bus mastering disk controller can transfer audio or video data directly from disk to the DSP subsystem without intervention from the host's CPU. Similarly, a bus mastering DSP subsystem can blast pixel data directly to a display's frame buffer. And, of course, the user gets the other PCI benefits such as plug-and-play automatic configuration, host platform independence and (finally) a robust and well-defined board-level interconnect for desktop computers.

DSP Backgrounder

Digital Signal Processing can be simply defined as the manipulation of analog signals that have been converted to digital values in order to perform a useful function. While the mathematical field of DSP has been around for centuries, it is only in the past 10-15 years that general-purpose DSP chips are available. These chips are similar to conventional microprocessors, but have a number of architectural features that boost the performance when performing DSP-related arithmetic. For example, a common DSP operation is to multiply a sequence of numbers by another sequence of numbers and then add (or accumulate) the products. On most general-purpose CISC and RISC processors, this requires a separate multiply and add instruction, plus any extra cycles for address register updates. Also, integer and floating point multiplies are multiple-cycle instructions in virtually all conventional CPUs. DSP CPUs can perform a multiply and accumulate all in a single instruction cycle and have other features making them well-suited to dealing with massive amounts of real-time data. Some of the characteristics of typical DSPs include saturation arithmetic (numerical overflows saturate to full-scale values rather than wrapping around), fast interrupt latency (to respond to real-time interrupts), one or more on-chip DMA channels, multiple external memory buses and parallel or serial communication ports (for interconnecting multiple DSPs and/or I/O channels).

Simpler DSP chips are so cost-effective that most mid- and high-line MPC audio cards incorporate one for music synthesis, mixing, equalization, level control and other functions. These DSPs can also be used for fax/modems, voice recognition, text-to-speech and general-purpose number crunching. However, this generation of DSP is generally inadequate for video processing at anywhere near real-time rates. For this, a more advanced architecture was required: The TMS320C80.

The Texas Instruments TMS320C80 DSP

The TMS320C80 DSP diverges from the (until this point) upward-compatible TMS320 series of digital signal processors with a new advanced architecture that combines four integer DSPs and a 32-bit floating-point RISC core on a single chip. The table below summarizes the key features of the C80.

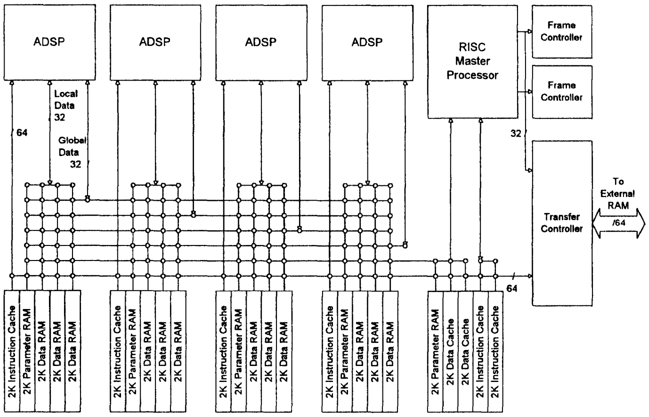
2 billion operations per second @ 50 MHz
Integrated onto a single chip:
Four Advanced Digital Signal Processors (ADSPs)
100 MFLOPS 32-bit RISC Master Processor (MP)
Transfer Controller (TC) — intelligent DMA controller
Two video memory Frame Controllers (FCs)
50 Kbytes of SRAM in 25 2K blocks
Memory crossbar switch supports 15 concurrent accesses per cycle
4 million transistors, 0.6 micron process
305-pin PGA, 7.5 W @ 3.3 V

TMS320C80 Key Features

The four ADSPs work in parallel with the RISC Master Processor and are fed data and programs through the memory crossbar switch to the 25 banks of internal memory. This memory is also accessible to the intelligent Transfer Controller which acts as a super DMA controller for moving data and programs on and off the chip. All told, the C80 can move 4.2 gigabytes of data per second within the chip plus up to 400 Megabytes/sec to off-chip memory.

The four ADSPs utilize a 64-bit instruction word to support multiple signal processing, bit-field/pixel manipulation and entropy encoding/decoding operations per instruction cycle. The 100 MFLOPS RISC CPU has a new architecture optimized to run code generated by high-level language compilers such as C. The RISC core has a separate floating point adder, multiplier and integer unit, all of which can run in parallel.

The dual frame controllers can interact with two video frame buffers simultaneously. These buffers can be connected to video capture and display devices. PAL, NTSC, interlaced and non interlaced as well as progressive scan formats are supported. The Transfer Controller directly controls the external memory connected to the C80 DSP. This memory can be SRAM, VRAM, DRAM or a combination of the above. The Transfer Controller also performs automatic endian conversion between big- and little-endian systems as well as handling dynamic bus sizing of 1, 2, 4 or 8 byte widths. Data transfer rates up to 400 Mbytes/sec are supported by the Transfer Controller between the C80's internal RAM and external RAM.



TMS320C80 Block Diagram

With a peak rating of 2 billion operations per second, the C80 can handle complex tasks well beyond the ability of the PCI host's CPU. The chart below lists some of the capabilities of a single TMS320C80 DSP.

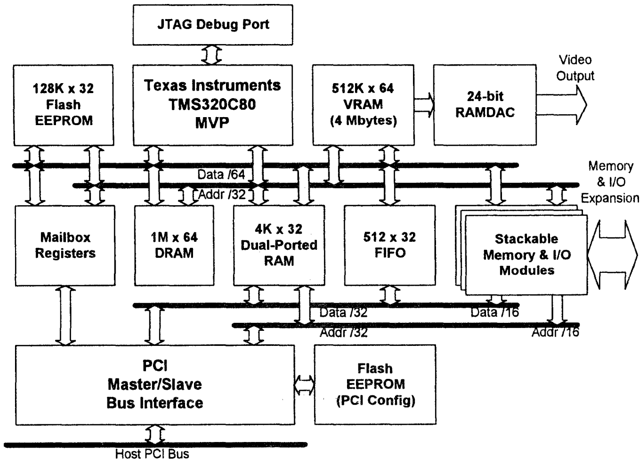
2 billion operations per second @ 50 MHz
130K Dhyrstone MIPS (MP alone)
JPEG encoding of 352x240 image (4:2:2 format) in 8 milliseconds.
Decoding of same image in 6 milliseconds
DCTs on 8x8 kernels at 800,000/second
700,000 shaded polygons/second
Px64 encoding and decoding at 30 fps
Real-time MPEG-1 encoding (CIF resolution)

TMS320C80 Capabilities

Another key issue to consider when comparing a software-only solution to a special-purpose DSP subsystem is that the DSP works in parallel with the host's CPU. While the DSP is working, the host CPU can be left free to attend to its normal functions of dealing with the host operating system, disks, display, keyboard, etc. This is especially important when time-consuming tasks are being performed, as using a dedicated, attached DSP can release the host computer for other uses concurrently.

The Griffin Board

The Griffin board mates a 50 MHz TMS320C80 DSP with memory, modular I/O and a high-performance master/slave PCI interface. As a PCI slave, a Griffin may be the target for host data transfers or from other bus master devices such as fast disk controllers. As a PCI master, a Griffin can (under control of the host) directly transfer data anywhere in system memory. Modules for NTSC video in and video-in-a window are planned initially, and others will follow. For example, the base Griffin board includes a single high-resolution RGB video output. With an additional channel of RGB output on an I/O module, the system is now capable of stereoscopic 3D graphics for virtual reality systems. Other stackable modules could include additional memory, parallel or serial I/O, audio I/O and other special-purpose interfaces.



Griffin Block Diagram

For dedicated, standalone applications, it is possible for Griffin to boot from its 128K x 32 nonvolatile EEPROM memory. This memory can also be used to store user or application preference data, error logging, serial number or other product information and proprietary algorithms.

There are three data paths from the C80 to the PCI bus, each optimized for a particular type of data transfer. The simplest is a set of mailbox registers that may be used to signal user-defined events. The mailbox registers may be accessed asynchronously with respect to other types of data transfers through the board. There is also a 4K x 32 bit dual-ported RAM placed between the C80 and the PCI bus. This RAM can be used for program loading, parameter and status passing and for quickly moving relatively small chunks of data. Finally, the bi-directional 512 x 32 bit FIFO is ideal for streaming arbitrarily large blocks of data on and off the board. The host (or any other PCI bus master) can set up the Griffin to automatically transfer data from anywhere in memory to the C80 and visa-versa.

Features	**Benefits**
Master/slave PCI Interface	High-speed 32-bit bus for highest host throughput
64-bit memory architecture	Maximize TMS320C80's bus bandwidth
8 MB DRAM memory space	Vast storage for programs and data
4 MB on-board frame buffer for RGB video output - 64-bit organization.	High-resolution output even in 24-bit color modes
Stackable I/O modules	Adaptable to various I/O requirements
Allows customizable I/O	Custom and embedded applications
Additional video output in RGB or NTSC/PAL/Component video formats	Flexible I/O configurations
512-word bi-directional FIFO and 4K x 32 dual-port RAM between C80 and host	High-speed data channel between the C80 and the host
512 Kbyte nonvolatile Flash memory	Standalone or embedded applications
JTAG port and on-board controller chip	Use familiar XDS debug tools

The on-board RGB output is backed with 4 Mbytes of VRAM and is capable of resolutions up to 1600 x 1280 with 64K colors. 24-bit color modes are available up to 1024 x 768 resolution. The high-performance RAMDAC includes a hardware overlay plane, dynamic color look-up tables in low bit-per-pixel resolutions and gamma correction tables in true color resolutions.

Debugging such a complex system might seem daunting as many of the board's resources are not directly accessible to the host, and there's the added complexity of five independent processing elements inside the C80 chip. Fortunately, the chip designers foresaw this situation and created a special emulation/debug port directly on the C80's silicon. This port is based on extensions to the IEEE 1149.1 JTAG standard for in-circuit testing of chip devices. The emulation interface permits complete control of the C80 and its peripherals and memory either through the PCI host (a JTAG controller circuit is included onboard Griffin) or through a remote debugger that attaches to the Griffin's JTAG port via a simple ribbon cable. While the C80's instruction sets are unique, the debug and development tools will be familiar to anyone with experience with the Texas Instruments XDS tools for the TMS320C3x, TMS320C4x and TMS320C5x series DSPs.

The TMS320C80 DSP on the Griffin board requires 3.3 volts rather than the more usual 5 volts for digital logic. PCI supports both 3.3V and 5.0V signaling, which is accomplished via keys in the edge connector. This makes it impossible to install a board into an incompatible slot. The C80 requires up to 7.5 Watts at 3.3 volts, but there is no guarantee that all PCI hosts will have sufficient 3.3V power available (or any 3.3V power at all, for that matter). So, the Griffin board includes a DC-to-DC converter to derive all of its power from the host's 5V supply.

Applications

The Griffin board has many applications in the fields of multimedia, general-purpose signal processing and number crunching, communications and military/industrial products.

Multimedia authoring
Real-time audio/video processing
Real-time and near-real-time compression
Graphics rendering
Virtual Reality
Spatial sound processing
Stereo image rendering
Teleconferencing
Real-time audio/video compression
Protocol conversion
Video-in-a-window
Commercial / Industrial / Military
General-purpose, high-performance signal processing
Medical image processing and display
Scientific visualization of complex data sets
Commercial-off-the-shelf DSP systems
Machine vision and real-time video processing
Embedded video processing systems
OEMs
Custom, stackable I/O and memory modules
Special configurations for high-volume applications
Graphics and image processing libraries

Griffin Applications

Summary

The PCI bus enables new applications for desktop computers when coupled with the TMS320C80 DSP. The Griffin board is well-suited for real-time video processing and has many potential applications in multimedia, teleconferencing and other areas. For special applications, the modular Griffin architecture supports multiple, stackable I/O or memory modules and the on-board JTAG emulation port makes in-circuit debugging a simple task.

The C80 and the Griffin bring the best of the new while retaining a path to the familiar. The faster and more robust PCI bus supplants the ubiquitous ISA bus and the C80's novel, new architecture is supported by an industry-leading manufacturer and familiar development tools. Griffin represents a leading-edge system that delivers the highest possible performance from both PCI and the C80.

PCI BASED MULTIMEDIA

Rainer Hoffmann
Thesys Mikroelektronik
Haarbergstrasse 61
99097 Erfurt, Germany

ABSTRACT

The paper covers the requirements of adding Multimedia capabilities to the PC platform. The PCI bus is discussed as a foundation for such capabilities and is compared to alternative architectures. Considering an application oriented approach, a list of Multimedia capabilities is compiled. Based on these prerequisites the design of a single chip Multimedia I/O chip is described.

THE TRADITIONAL WAY

Conventional wisdom says that if you are doing a Multimedia card, make it an ISA one. There are two reasons why ISA has been and still is the bus for just about every Multimedia card in the known universe.

1: ISA slots are in every PC and everybody wants the biggest TAM available.

2: Multimedia cards, such as sound cards or simple video capture cards, either did not require high bandwidth from the bus or they worked their way around the limitations of the ISA bus.

Things are changing, though. These changes are driven by the arrival of video as a Multimedia data type. When video first arrived on the desk top, designers had no choice but to use their own special interface just to display full rate video on a monitor. In those days one would not begin by defining an industrial standard and then design the products. As a result, designers were stuck with an oddity called the feature connector. And god knows they have paid for their negligence. Besides incompatibilities, the major drawback has been that the video data stream bypassed the system and went directly to the monitor. This architecture increased the system cost more than it should and solved only part of the applications requirements. This kind of architecture could not handle situations effectively where applications would want to process the video, nor could this kind of interfacing support more than one source of the video. Hardware compression was the only way of making even VHS quality video data available for processing. The additional cost of the hardware compression limited the appeal of such solutions to the broad market. Today we have high performance CPUs and high speed hard disk systems that could handle higher data rates directly. It is probably not wrong to assume that in the future even more powerful standard system components will allow more economical video solutions, if there is a way of getting the video to them.

If video is to become a part of just about every PC, things have to become less expensive and video has to get to every system component, not just the monitor.

AS GRAPHICS GREW UP

Graphic chip manufacturers have been approaching the issues of bus bandwidth for some time. The true performance revolution has come from the use of local buses as the system interface. The commercial and technical ratio behind local buses has moved the entire PC market to adapt these buses. As video can be considered a special case of graphics, the same ratio can provide significant benefit to video applications. One interesting side note may be added here. The first widely used local bus has been the VL bus. This bus was defined by VESA, an organization who's primary interest is in graphics. The VL bus has many benefits to the PC market; still it has been quickly realized that focusing on the requirements of the graphic system alone will not necessarily provide a good solution for the entire system. The radical swing of the market away from VL towards PCI has proven that a well defined interface, which provides benefits to the entire system, will provide enough economical benefit to change the market in its favor.

THE MULTIMEDIA APPLICATION

There is of course no such thing as the Multimedia application. Still we can make a list of things that a typical Multimedia application will use. The point I am trying to make is that if we were to conceive the theoretically best Multimedia component, this component should be able to cover all of the required capabilities. As of today, when we open a Multimedia ready PC we will find multiple cards that all solve their individual part of a Multimedia applications task. One card offers the sound features, another one will capture and overlay video and yet another will do MPEG decoding. The modem function will occupy another card or reside outside. As we move into the age of the Multimedia PC for just about everybody, such variety becomes a commercial disaster.

Let's make a list of functions that are part of a Multimedia application. First of all, we need sound. Sound will be generated by the PC and may also come from a prerecorded medium such as a CD. We will also need video. Again, with the entrance of MPEG we will see prerecorded video and graphics data generated by the CPU. One requirement is to combine the PC generated componentwith the prerecorded components. For audio this boils down to an audio mixer. For video this dictates the need for at least a pixel resolution mixing of video with graphics and a way of displaying the result.

The more tool-like Multimedia application such as video phones require efficient ways of moving the Multimedia data into and through the system. Possible tasks range from capturing to getting the compressed data to the decompression hardware.

One way of getting all these function on one board would be to put everything into a single chip. Another way could be to build chip that would work as a bridge between a system bus and subsystems on the board. Actually it is not that important how many functions are integrated into the chip. The point is that it has just one system interface. Such a chip is similar to a Multi-I/O chip. That is why we called it a Multimedia Multi-I/O chip.

HOW TO INTERFACE A MULTIMEDIA MULTI I/O

The number of possible interfaces is definitely higher than one, so lets reduce the number of candidates one by one.

The one major headache for hooking up video is bandwidth. The nasty thing about video is it comes into the system in real time. The incoming data has to be processed at the rate of the incoming video data. One trick to reduce the bandwidth demand is to scale the video to a smaller format. The required average bandwidth goes from about 1 MB/s (that is M for 10E6) for a 160x120 format at 30 frames per second (fps) with 16 bit color depth to about 33 MB/s for a full size PAL format (768x576 @ 25 fps) using true color format. Table 1 demonstrates how the bandwidth requirement goes up with the image size. The required peak bandwidth varies, of course, with the size of your FIFO. If one would try without a FIFO the peak bandwidth would even reach 44 MB/s for full format PAL at 25 fps in true color.

# horz. pixels	# vert. pixels	fps	pixel/s	MB/s @ 16 bit color depth	MB/s @ 24 bit color depth
160	120	30	576.000	1	2
320	240	30	2.304.000	5	7
352	240	30	2.534.400	5	8
640	480	30	9.216.000	18	28
384	288	25	2.764.800	6	8
768	576	25	11.059.200	22	33

Table 1. Average bandwidth requirements over image format

There go ISA and EISA, and on the little more exotic architecture side we can also exclude: RS232, SCSI, IDE, Access.bus and, of course, the keyboard and PS/2 mouse interface. I presume we would like to use an electrically clean and well-defined bus. Therefore we can add to the list of 'you shouldn't even think about it' good old VL-bus and the all time favorite, the popular feature connector.

Let's see! We have come down with three candidates for the title: VAFC, VMC and PCI.

VAFC provides the designer with a technically sound and very economical way of displaying video on a graphic screen. Still, all video subsystems require a second bus interface to the system for doing things like configuration and capturing. As a reaction to this issue, companies have come up with their own special flavor of VAFC. This common approach of fixing things that were left out of a standard has in the past provided us with many delightful defacto non-standards. Even those special VAFCs are still not capable of handling sound, nor are they a likely candidate to hook up a DSP to your system. The biggest issue with VAFC is: it isn't even a bus. It's a point to point connection between one video card and one graphics card. Try to get yourself a capture card and a MPEG card and hook those up via VAFC. You should leave the computer case open, because you have to switch cables when you want to see an MPEG video after having captured some pictures for the family album. Sorry, but we only have two candidates left.

VMC was defined as a bus to handle video and all the other data types required for the tasks described above. It is technically sound and quite capable. The only issue not completely clear is how do we get data to and from the rest of the system? We have two choices, either to do all communication through the graphic controller or get a bridge between a Multimedia capable system bus and VMC. As we have only two candidates left over I should probably have written PCI instead of 'Multimedia capable system bus'. The everything through the graphic card approach is not especially attractive to graphic chip designers that are trying to squeeze every single byte per second of bandwidth out of their memory interface just for display refresh and fast drawing. The audio and compression related data transfers could take a major byte out of the available bandwidth. Since upcoming things like 3D are not likely to reduce the bandwidth hunger of the generic graphic card tasks, we better try to do it the other way.

To be completely honest, if this would have been written in January 94, this would be a paper about a VMC to PCI bridge. The architecture is quite appealing. VMC reduces the bus load on the PCI to only the necessary part. Simply displaying a video picture would not take a single megabyte of the PCI bandwidth. The only problem is that you need a graphic controller that has a VMC interface in your system. Many people have announced VMC graphic controllers, none has been seen yet. The reason for this is partially the old chicken and egg problem. Who would deliver video on VMC to my VMC graphic controller, if I were to bring one to the market? For sure this problem will be resolved as people are teaming up to finally get things going. The second reason is more serious. It can be summed up by a single number: 44. This is the number of pins that you have to add to your graphic controller to have a 32 bit VMC interface. As things are moving towards 0.5μ and even further, being pad limited becomes a serious concern. Pad limited refers to situation where your chip has to be bigger than the logic that you want to put on the chip would require because you have too many pins. A few semiconductor manufacturers offer special technologies to fix this issue. Still, the most common approach is just not to have too many pins. As a result of this discussion, it is probably not wrong to assume that not every graphic controller will have a VMC interface. This of course limits the TAM for any VMC based solution and is just a nag. Our guess is that VMC can be a choice for optimized systems, but the main bulge of the business will not use it.

AND THE WINNER IS - PCI

It seemed to be a natural choice to make some good remarks about PCI at a PCI conference. So it is my pleasure to declare PCI as the Multimedia bus. The ratio seems straightforward:

a) PCI provides the bandwidth,
b) the system wide access and
c) the biggest TAM available,
when compared with alternatives that provide a) and b).

On the negative side of things are uncertain bus latencies, unguaranteed bandwidth into system components, and the concern that the Multimedia traffic will clobber the entire bus. The latency problem relates directly to the required FIFO depth, since you have to buffer the incoming video data until you get access to the bus. At this point, it is time to thank the guys at Intel for their pioneer work done on sending video over the PCI bus. The result of their efforts is now being sold by Philips as the SAA7116. Their experience, that they very generously share, has been that you can configure most systems in a way that makes Multimedia traffic possible. But the issue is not to be underestimated. Very many parameters are involved in finding what the real performance of a given PCI system for Multimedia purposes is. For the sake of the market there should be a set of measurements that can identify a system as perfect, good or maybe even unusable for PCI based Multimedia. This would allow manufacturers to put a PCI Multimedia performance requirement statement on each product, just as we have the hardware and software requirements today. If the industry decides not to hurt anybody and avoid súch benchmarks, it will not be to the benefit of PCI as a Multimedia bus and, thus, not for the benefit of the industry itself.

For the time being let's stick with the statement that PCI can hold Multimedia traffic and explore the potential of this statement.

SCALER2/PCI - A MULTIMEDIA MULTI-I/O

Since 1993 we at Thesys offer Multimedia solutions that integrate video into the PC environment. As we strongly believe that Multimedia products have the same price pressure that their consumer counterparts face, we have been sticking from the start with the shared frame architecture. The only issue that we have with the current implementations is that we have to provide a separate chip for every graphic controller that we want to support. Hooking the video processor via PCI to the graphic controller gets rid of that pain. PCI also provides us with an opportunity to build a part that eventually will be able to handle more Multimedia tasks in a useful way.

The primary task for our Scaler family is to take data that comes into our chip as digital video data, filter it, scale it and do some color space conversion on it. The second thing is to take that video data and display it. Displaying it is more than just switching to full screen video or having the video run by itself in a window. MPEG, for instance, is and will be used as prerecorded video data that interacts in some way with PC generated graphic as a means of user interaction. A simple example of this is a Karaoke title that contains multiple languages from which the user can choose. The text will be generate by the CPU. As experts in this field will assure you, it is absolutely necessary that the text will be displayed within the video. Therefore we need a mechanism to mix the video with the graphics. This sounds easy enough, but it brings up several issues. Traditionally, you would use an overlay architecture, have dedicated (additional, wasteful, ...) memory for the video, and use color keying to replace every graphic pixel of a certain color with the corresponding video pixel. Now that everybody wants to save money, the idea of using a shared frame buffer is becoming more popular. The shared frame buffer architecture gets rid of the second memory system for video by storing the video data inside of the graphics memory, effectively treating it as normal graphic data. But beware, mixing video and graphics in a shared frame buffer architecture is not as easy as some people think. It takes some care to guarantee that the refresh of the video and the refresh of the graphics are synchronized. A simple mask that decides whether to overwrite or not to overwrite a graphic pixel by a video pixel will work fine with stationary graphic overlays, but might provide some undesired effects when the graphic object starts to move. Fixing this issue requires full access to the memory where the graphic comes from. This is traditionally the memory of the graphic controller. As new graphic architectures come to the market, other places might become the source of the data. For instance WinG, the Microsoft game API, will render in main memory and copy the entire screen to the graphic memory when the drawing is finished for that frame. This could also need some help, could it not? The Scaler2/PCI will actually take data from anywhere in the system, merge that data with the incoming video, do some processing and send it to the graphic memory, all under the timing regime of the video. This effectively eliminates any chance for video and graphic to get out of sync and therefore eliminates all related artifacts. The bandwidth requirement of such a configuration will put a very heavy burden onto the PCI bus. Our calculations show that a good PCI system will be able to handle this. Still as an architectural element that can significantly off-load bandwidth from the bus, Scaler2/PCI has a memory interface for optional memory. This memory could also be used by the second major functional block: the DSP interface.

INTERFACING A DSP VIA PCI

The primary motivation to put a DSP into a PC system is for audio purposes. Audio comes in many flavors. There is Blaster type sound card audio, wave table audio, modem and fax audio. A single DSP based audio solution can cover all of these applications. This has been demonstrated by DSP-based sound cards already. The potential of using a DSP has been increased by Microsoft with the introduction of the DSP platform. This software interface will make DSP capabilities available to Windows95 applications. Obviously the hardware architecture has to support these efforts. Putting the DSP on an ISA interface is a very efficient way to clobber system performance when the DSP is receiving or sending data. This becomes a real tragedy for systems that have a PCI bus. The ISA bus in these systems is implemented by a PCI to ISA bridge. Pushing the ISA bus to its limits will also block the PCI bus. This is what I call a waste. In contrast, the interface we have chosen is a PCI master providing to the DSP a DMA type access to and from the system. The bandwidth requirements of full CD quality audio can be satisfied with a small percentage of the total capacity of the PCI bus. This interface could also serve to send very high bandwidth data to the DSP. Now that software has finally realized the power of DSP computing, hardware shouldn't be in the way of new DSP applications; one of which is obviously MPEG audio en- and decoding.

HOOKING UP MPEG

MPEG will play a major role for everybody that is involved in PC based video. The MPEG market is so promising that people come up with new ways to get video into the PC. Besides doing it the old fashion way with an overlay controller, there have been some proprietary interfaces introduced lately. From a hardware designers point of view there are two basic task to be solved for a MPEG card design. First you must get the compressed video stream to the MPEG decoder. Some newer decoders provide direct CD drive interfaces, which makes perfect sense since they have been designed with standalone consumer type products in mind. For a PC this kind of interfaces will work with most applications we have today, but what if we get this thing called the information superhighway? In a PC environment we will see MPEG coming from almost anywhere in the system, not just the CD-ROM. Now, why not let the CPU pump the data to the MPEG decoder? The answer to this is that there is a better way of doing it. A PCI master controller could transfer the data. By integrating this function with other functions, such as the video data transfer and the DSP data transfer, this could even be achieved without a heavy investment in silicon real-estate. This would off-load the CPU and make the playback task more multitasking friendly. In other words, it would provide an efficient solution for things to come, one of which will be MPEG2. MPEG2 will raise the amount of data to be moved to the decoder from about 150 KB/s to anywhere from that amount to more than 1 MB/s. Anybody that has tried to satisfy a hungry sound card with CD quality 44 kHz/16 bit stereo audio, which is about 176 kB/s, knows that even this modest requirement can be a problem, especially if you try to do this while running a certain operating system on your machine at this time.

The second task is to display the decoded video data. As this data is of the same format as live video data it can be treated just the same.

THINGS TO COME

In the future Multimedia applications will call for new features and for less expensive implementation. Topping of the list of new features for us is 3D graphic support and interaction between video and 3D graphic. Using the Multimedia MULTI-I/O as our basic architecture, we will add 3D capabilities to the system.

MPEG decoding in software is currently making waves. Today even the most powerful PC's have just enough horse power for basic decoding, if there is some hardware that helps to do part of the job. We are focusing our design efforts on a generalized form of this cooperation between existing processing power and helping hardware. Hardware-software co-design tools help us raise the performance of a system for a given task above the required limit. This kind of approach is also at the core of our cost reduction efforts, which will be focused on using the DSP for things beyond the basic audio tasks described above.

IT'S ALL IN THE GRAPHICS CARD

The tight coupling of video and graphic has brought about a new class of graphic cards that include Multimedia functions on the graphic card itself. These cards are less expensive and provide better quality to the customer than two board solution. The movement towards those integrated solution is accelerated by graphic chip manufacturers that develop special architectures around their graphic solutions. Most of theses concepts solve the display of the video issue, some are trying to cover audio as well. The one issue none of these special architectures approaches is the upgrade issue. True, we can use more gates today than we could before, and also true the know-how to process Multimedia data is much more wide spread than it use to be. As a result we can do many new things. Some people even believe they can do everything. Still, the one thing we will never be able to do is to fix tomorrow's problems. That's why PCI is a much better choice as the Multimedia interface than optimized proprietary interfaces. Integrated and cost optimized solutions will only be of true value to the customer if they make their resource available through the PCI bus. If they meet this requirement they are open for future upgrades.

AN OEMs VIEW OF MULTIMEDIA

As Thesys is not an OEM, we can only speculate on what the view of an OEM could be. I think it is safe to say that OEMs prefer to have the solution that costs the least in its category. Probably the OEM also prefers to have the smallest possible number of different Add-On cards. Also it is wise for the OEM to be flexible in the configuration of their systems, in order to be able to cover the varying demands of the end customers. The last two points seem to be contradicting. Still, it is possible to satisfy both targets by finding or defining elementary tasks that simply belong together. I hope to have shown that a PC that is to be called a video ready Multimedia PC requires both audio and video and the corresponding merging functions. Given that you can build a single PCI card that covers all of these functions, it would give the OEM the option to take a plain vanilla motherboard and turn it into a Multimedia PC with just one board. Since the proposed solution, the Multimedia Multi-I/O, uses the least overhead and the most cost effective architecture to supply these features, it will also meet the number one requirement:

> The most bang for the buck.

PCI based Graphics board for high performance GUI acceleration and 3D graphics

Chris Russell
Marketing Manager, 3-D Graphics
Cirrus Logic
3100 W. Warren Avenue
Freemont, CA 94538
Ph. (510) 226-2358 Fax (510) 252-6070

The advent of high performance graphics chipsets such as the Cirrus Logic GD5470/71/72 enables a new breed of Pentium/PowerPC machines with PCI to achieve the performance traditionally associated with the workstation market.

The presentation will cover the emergence of the 3D based PC market, the technologies such as PCI that are making it possible and also the specifics of the Cirrus Logic Chipset.

A PowerPC Add-On Graphics Board

Ken Comstock
Diamond Multimedia Systems, Inc.
2880 Junction Ave.
San Jose, CA 95134
(408) 325-7000/7070 (fax)

The PowerPC architecture requires different add-on boards than those suited to the general PC market. A new graphics board from Diamond Multimedia System illustrates the requirements and points the way to further developments in this area.

OPEN FIRMWARE AND PCI

Mitch Bradley
FirmWorks
480 San Antonio Road, Suite 230
Mountain View, CA 94040-1218
wmb@firmworks.com

ABSTRACT

IEEE Standard 1275-1994 for Boot Firmware (based on Sun's "Open Boot" firmware) was approved in April, 1994. The specification defines the boot firmware more commonly known as "Open Firmware".

A group of companies is proposing the use of Open Firmware for PCI Local Bus add-in cards, to eliminate the need for a different boot driver for each different CPU type.

Open Firmware is a CPU-independent, bus-independent, and operating-system-independent specification for the firmware that lives in main system ROMs, with the following features:

- Autoconfiguration of add-in cards on multiple buses
- CPU-independent booting from add-in cards
- CPU-independent selftest for add-in cards
- Built-in debugging tools for hardware, software, firmware
- Extensible NVRAM configuration options management
- Built-in scripting language for patches and tests
- Autoconfiguration assistance for operating systems

Open Firmware is based on technology that has been shipping for many years, on over 1 million systems. Compatible implementations of Open Firmware are currently available for a number of different CPUs.

This presentation gives an overview of Open Firmware and its application to the PCI Local Bus. The talk includes both technical information and the business case for firmware standardization.

BIOGRAPHY

Mitch Bradley is president and chief technical officer of FirmWorks, a consulting firm specializing in providing Open Firmware system ROMs, device drivers, and training. Prior to founding FirmWorks in February, 1994, Mitch was the technical leader of the firmware group at Sun Microsystems Computer Corporation, where he conceived and developed the Open Firmware concept. He is chairman of the IEEE P1275 Open Firmware Working Group, which developed the Open Firmware standard, and vice chairman of the ANS X3/J14 Technical Committee, which developed the ANSI Forth standard. His educational background includes degrees in various technical disciplines from Vanderbilt University, Cambridge University, and Stanford University. Before embarking on the quest for Open Firmware, he did a variety of electronics things, from analog circuit design to operating system hacking.

IBM Perspectives on PCI's Use with its PowerPC Products

Lee H. Wilson
IBM Power Personal Systems
Mail Stop 4357
11400 Burnet Rd.
Austin, Tx 78758
Phone: 512-838-6569
E:Mail: leew@austin.ibm.com

IBM has made it clear that PowerPC is our strategic processor technology. Our strategy for PowerPC is clearly a multi-OS strategy. PowerPC will become a unifying theme throughout our product family as our strategies continue to unfold. We are committed to using PCI. PCI provides for the support of multiple processor architectures. We have had a number of discussions with the PCI development community on issues that have arisen in supporting this open environment. Firmware issues, big and little endian issues and processor bus issues arise frequently since we have some different requirements in these arenas. We have identified other areas of interest that we would like to bring to the industry's attention as well.

Our presentation will touch on the following subjects:

- Our reference platform for PowerPC clients - today and tommorrow.
- Bi-Endian issues
- Should we use PCI-based graphics or processor bus-based graphics?
- Where is PCI applicable to multi-media and where is a real time bus like VMC applicable?
- Our move to open firmware.
- What are we doing about runtime firmware support?
- PCI error recovery
- Do we need new arbitration schemes that support real time requirements?
- Implications of multiple host bridges in a system
- 64 bit addressability
- 3rd party data moving mechanisms

INTRO TO THE HP PCI BUS EXERCISER

Tom Shanley
MindShare, Inc.
2202 Buttercup Drive
Richardson, TX 75082
Tel: (214) 231-2216
Fax: (214) 783-4715
Email: mindshar@interserv.com

Abstract

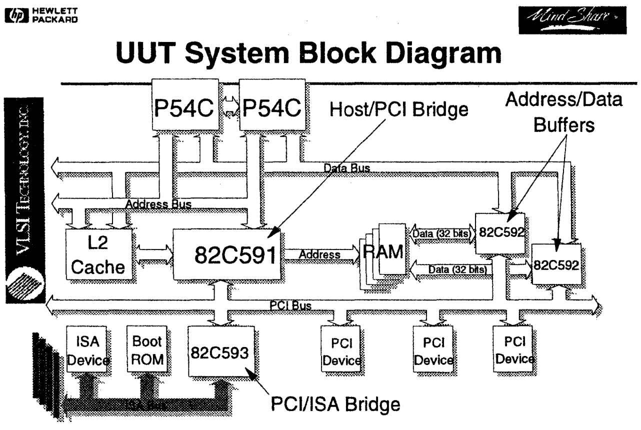
This paper provides an introduction to the HP PCI Bus Exerciser. The exerciser may be utilized either as a passize observer or as an active participant. In the role of passive observer, it captures and verifies the proper protocol of PCI bus traffic originated by other PCI bus masters. As its name implies, the exerciser may also be used to generate PCI bus transactions. It can act as either the bus master or as the target of PCI transactions.

This paper identifies the key hardware and software components of the exerciser system and their relationships to the UUT.

TYPICAL SYSTEM ARCHITECTURE

Slide 1 illustrates the architecture of a typical PC system that incorporates the PCI bus. The example system is built around three buses: host, PCI, and ISA. The host processor and bus masters residing on the PCI and ISA buses perform read and write transactions to transfer data with targets residing on the same bus or a different bus. The VLSI 82C59x chipset provides the bridging between the three buses.

Slide #1



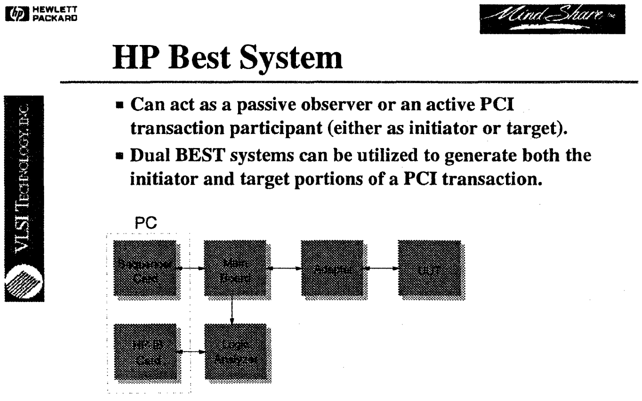
UUT System Block Diagram

HP BUS EXERCISER COMPONENT OVERVIEW

The HP PCI Bus Exerciser consists of both hardware and software components. Slide 2 introduces the hardware components. They are:

- Adapter — Interfaces the logic analyzer and Main Board to the UUT.
- Main Board — Executes test sequences downloaded from the Sequencer Card and returns results to Sequencer.
- Sequencer Card — Compiles test sequences and downloads to Main Board for execution. Receives results from Main Board.
- Logic Analyser — Captures bus traffic and downloads to test system.

Slide #2



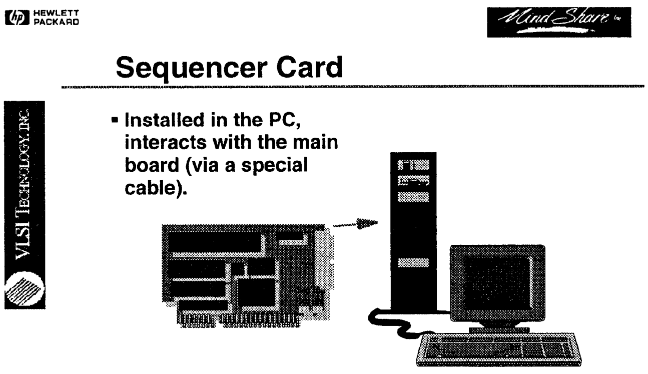
HP Best System

- Can act as a passive observer or an active PCI transaction participant (either as initiator or target).
- Dual BEST systems can be utilized to generate both the initiator and target portions of a PCI transaction.

SEQUENCER CARD

The Sequencer (ISA) card is installed in the PC. It runs the sequencer program and interacts with the Main Board's pattern recognition hardware to control the execution of the specified sequence. Slide 3 illustrates the Main Board and the connector used to interface with the Sequencer card.

Slide #3



Sequencer Card

- Installed in the PC, interacts with the main board (via a special cable).

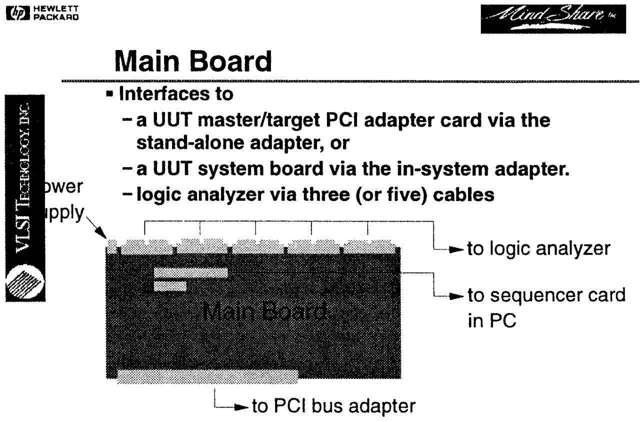
Main Board

The Main Board contains the bus protocol state machines, transaction memory and triggering logic. It interfaces with:

- the sequencer card and the PCI bus adapter that connects to the UUT's PCI bus.
- the logic analyzer. This connection permits the logic analyzer to observe the PCI bus and internal state machine signals on the Main Board without presenting an additional signal load to the PCI bus under test.

A separate power supply is connected to the Main Board.

Slide #4



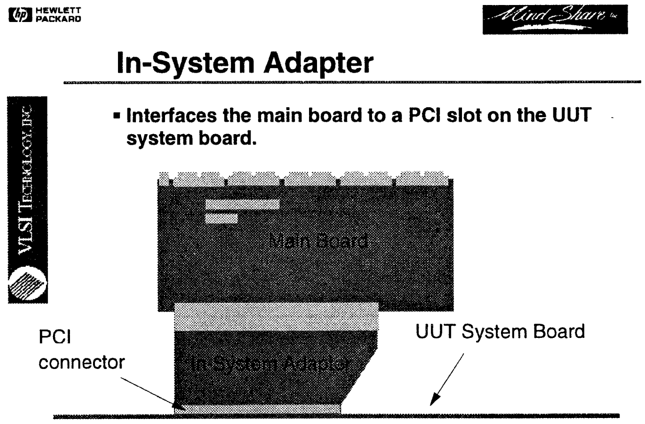
Main Board

- Interfaces to
  - a UUT master/target PCI adapter card via the stand-alone adapter, or
  - a UUT system board via the in-system adapter.
  - logic analyzer via three (or five) cables

In-System Adapter

In a scenario where the HP system is to interact with a UUT consisting of a PCI system board, the Main Board is interfaced to the UUT system board via the HP E2911 In-System Adapter.

Slide #5



In-System Adapter

- Interfaces the main board to a PCI slot on the UUT system board.

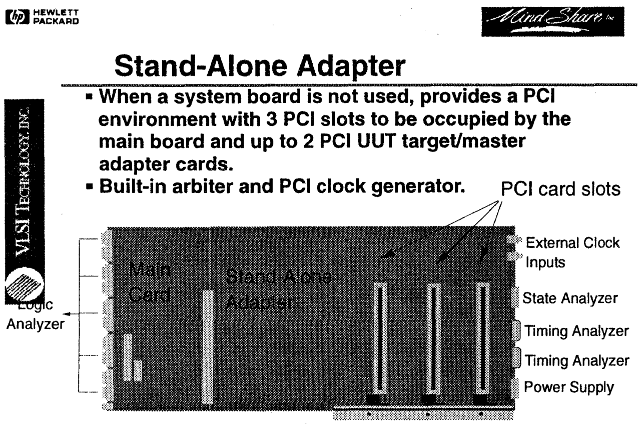
Stand-Alone Adapter

In a scenario where the HP system is to interact with a UUT consisting of one or more PCI expansion cards (rather than a PCI system board), the Main Board is interfaced to the UUT via the HP E2912 Stand-Alone Adapter.

The E2912 adapter is, in effect, a PCI System Board. It incorporates a PCI bus arbiter, three PCI card slots and a PCI clock generator. It permits the use of an external clock in lieu of the on-board clock.

The Main Board is mated to the HP E2912 Stand-Alone Adapter as an extension in the same horizontal plane. The logic analyzer and the Sequencer interface to the Main Board. UUT PCI cards are installed in the three card slots supplied on the adapter. Connections are provided for one external state analyzer and one or two timing analyzers. A separate power supply is connected to the adapter.

Slide #6



HEWLETT PACKARD MindShare Inc.

Stand-Alone Adapter

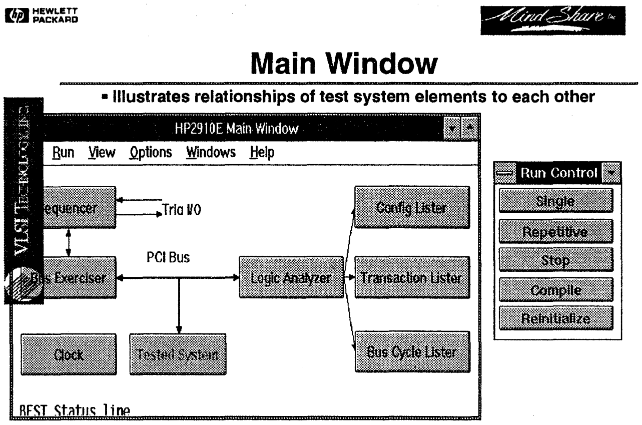
- When a system board is not used, provides a PCI environment with 3 PCI slots to be occupied by the main board and up to 2 PCI UUT target/master adapter cards.
- Built-in arbiter and PCI clock generator.

Main Window

Slide 7 is used to discuss the various elements of the main window. These elements are:

- Sequencer button. Invokes the sequence editor.
- Bus Exerciser button. Invokes the bus exerciser window.
- Logic Analyzer button. Invokes the logic analyzer window.
- Configuration Lister button. Invokes the configuration lister window.
- Transaction Lister button. Invokes the transaction lister window.
- Bus Cycle Lister button. Invokes the bus cycle lister window.

Slide #7



HEWLETT PACKARD MindShare Inc.

Main Window

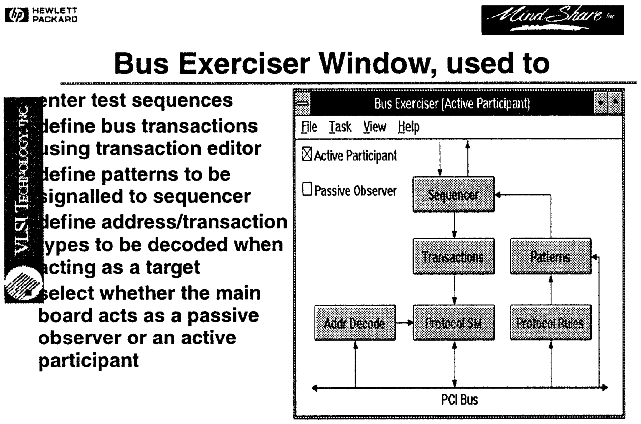
- Illustrates relationships of test system elements to each other

Bus Exerciser Window

The bus exerciser window is used to:

- enter sequences using the sequence editor.
- enter PCI transactions using the transaction editor.
- enter patterns using the pattern editor.
- enter address information to be utilized by the main board's address decode logic when the main board has been configured as a target device.

Slide #8



HEWLETT PACKARD

MindShare Inc.

Bus Exerciser Window, used to

- nter test sequences
- lefine bus transactions ising transaction editor
- lefine patterns to be ignalled to sequencer
- lefine address/transaction ypes to be decoded when cting as a target
- elect whether the main board acts as a passive observer or an active participant

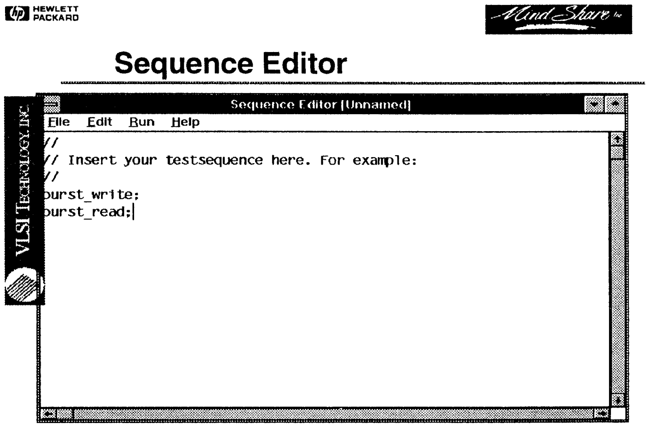
☒ Active Participant

□ Passive Observer

Sequence Editor

The sequence editor is used to enter test sequences. A test sequence contains calls to transactions defined in a transaction file and to patterns defined in a pattern file. It also contains control statements to control sequence execution.

Slide #9



HEWLETT PACKARD

MindShare Inc.

Sequence Editor

```
//
// Insert your testsequence here. For example:
//
urst_write;
urst_read;
```

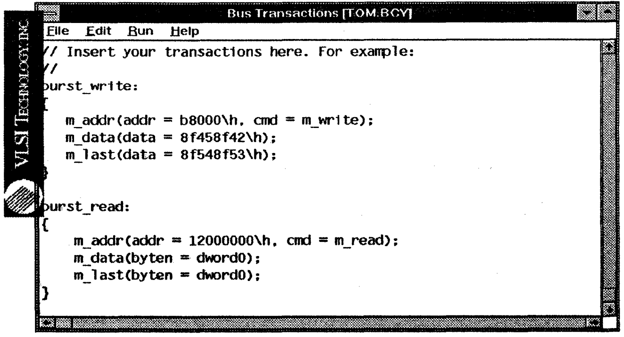
Transaction Editor

The transaction editor is used to define PCI bus activity to be generated by the main board. Each transaction is named and is called from within a sequence file by an INSTR (instruction) statement.

When executed, the examples in slide 10 cause the main board to perform a two data phase burst write or burst read transaction.

Slide #10

Transaction Definition



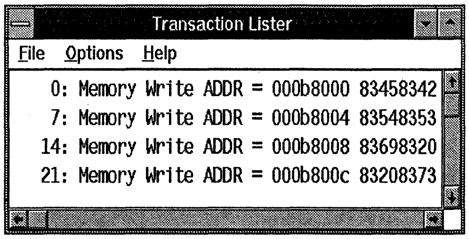
```
// Insert your transactions here. For example:
//
burst_write:
{
   m_addr(addr = b8000\h, cmd = m_write);
   m_data(data = 8f458f42\h);
   m_last(data = 8f548f53\h);
}


burst_read:
{
    m_addr(addr = 12000000\h, cmd = m_read);
    m_data(byten = dword0);
    m_last(byten = dword0);
}
```

Transaction Lister

The transaction lister window displays the PCI bus activity uploaded from the logic analyzer on a transaction-by-transaction basis. The bus cycle lister window may be used to view each transaction using clock-level granularity.

Slide #11

Transaction Lister



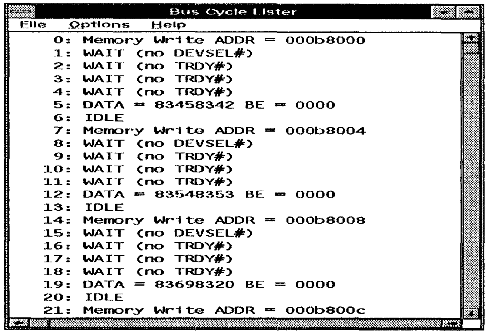
```
 0: Memory Write ADDR = 000b8000 83458342
 7: Memory Write ADDR = 000b8004 83548353
14: Memory Write ADDR = 000b8008 83698320
21: Memory Write ADDR = 000b800c 83208373
```

Bus Cycle Lister

The bus cycle lister window displays PCI bus activity captured by the logic analyzer on a clock-by-clock basis. The transaction lister window collapses this information to the lesser-granularity level of the transaction-by-transaction basis.

Slide #12

Bus Cycle Lister



```
 0: Memory Write ADDR = 000b8000
 1: WAIT (no DEVSEL#)
 2: WAIT (no TRDY#)
 3: WAIT (no TRDY#)
 4: WAIT (no TRDY#)
 5: DATA = 83458342 BE = 0000
 6: IDLE
 7: Memory Write ADDR = 000b8004
 8: WAIT (no DEVSEL#)
 9: WAIT (no TRDY#)
10: WAIT (no TRDY#)
11: WAIT (no TRDY#)
12: DATA = 83548353 BE = 0000
13: IDLE
14: Memory Write ADDR = 000b8008
15: WAIT (no DEVSEL#)
16: WAIT (no TRDY#)
17: WAIT (no TRDY#)
18: WAIT (no TRDY#)
19: DATA = 83698320 BE = 0000
20: IDLE
21: Memory Write ADDR = 000b800c
```

Configuration Lister

The author apologizes for not having a picture of the Configuration Lister screen. This lister decodes and lists each configuration transaction as a separate lien item. On each line, the following elements are identfied:

- PCI device number.
- Command type (read or write).
- Type 0 or 1 configuration transaction.
- Target PCI function number.
- Target PCI configuration register.
- Value read or written.
- If configuration register defined by specification, provides detailed breakdown (e.g., BIST implemented, cache line size).

Conclusions

The HP bus exerciser provides the following capabilities:

- controlled generation of the PCI bus activity initiated by both bus master and target devices.
- automatic protocol checks performed on all PCI bus activity generated by the bus exerciser, as well as the activity generated by PCI agents other than the bus exerciser.
- permits recording, editing and playback of PCI bus traffic.

In conclusion, the HP bus exerciser is an invaluable tool designed to ease PCI device design and debug.

Slide #13

Conclusions

- **controlled generation of the PCI bus activity initiated by both bus master and target devices.**
- **automatic protocol checks performed on all PCI bus activity generated by the bus exerciser, as well as the activity generated by PCI agents other than the bus exerciser.**
- **permits recording, editing and playback of PCI bus traffic.**

Author

Tom Shanley is president of MindShare, Inc., a supplier of engineering-level training to the design community. He formed MindShare 7 years ago to provide PC-architecture related training to both the hardware and software design communities. Since then, he has developed and continuously delivered courses on the following subjects:

- ISA System Architecture
- EISA System Architecture
- Micro Channel Architecture
- PS/2 System Architecture
- i486 System Architecture
- Pentium System Architecture
- PCI System Architecture
- PowerPC System Architecture

He has authored and self-published seven books describing the subjects listed above. Addison-Wesley will re-publish the entire book series in Q1 '95. The Pentium™ and PCI books were developed at Intel's request so that MindShare would be available to teach these subjects to Intel's internal design community and Intel's customer base on an on-going basis. Likewise, the PowerPC™ architecture book was developed at IBM's request. *ISA System Architecture* has been serialized in Japan's leading PC magazine, "Super ACSII", for the past 1.5 years. PC Magazine gave an excellent review of the entire book series in the July '94 issue.

During the past 7 years, MindShare has provided on-site seminars to its customer base. MindShare's customer base includes: IBM Corp., Motorola, Hewlett-Packard, Unisys, National Semiconductor, NEC, Compaq Computer Corp., Dell Computer Corp., Cirrus Logic, SCO, and VLSI.

Current Activities

Tom is currently engaged in the following activities:

- updating the entire book series for the Addison-Wesley launch.
- production of two new book titles for a spring introduction.
- continued teaching at customer sites world-wide.
- collaborating with Hewlett-Packard on the launch of the PCI Bus Exerciser.

SYSTEM-LEVEL EMULATION ENABLES PCI RAPID DEVELOPMENT

Vincent Coli and Doug Kern
Aptix Corporation, 2890 North 1st Street, San Jose, CA 95134

ABSTRACT

This paper describes a system emulation methodology for PCI rapid development. System emulation is performed on a reprogrammable hardware platform integrated with high-level design tools and populated with reprogrammable FPGAs. System emulation encompasses emulation of both embedded core ASIC(s) and the board-level components—together on one hardware platform. This robust methodology offers observability, changeability, and co-design.

Observability eases testing and design debug utilizing an HP logic analyzer. Software routable diagnostic probes isolate trouble areas in the circuit before they become problems.

Changeability empowers the designer to explore various architectures by implementing Verilog/VHDL and schematics quickly using programmable interconnect technology and reprogrammable FPGAs. Thus performance-stealing bottlenecks can be removed or designs can be reused for new projects.

Co-design enables an entire design team to participate in system-level testing of the hardware, software, bus interface, and human interface early and concurrently in the design process. Bus compliance can now be ensured before the design is committed.

INTRODUCTION

The PCI Local Bus was defined to establish a standard high-speed interface between peripheral functions and the system processor. Although the PCI Local Bus standard was defined to allow trouble-free connection of high-speed peripherals to the host system, compatibility problems do occur. System emulation catches compatibility problems very early in the design cycle.

Moreover, the variety of high bandwidth functions (e.g., graphics/multi-media, SCSI, IDE controllers, communications, high-speed LANs) configured around the PCI Local Bus is growing rapidly requiring integration testing with multiple add-on cards. System emulation can provide full system verification.

Finally, many of today's PCI-based high bandwidth functions involve real-world interfaces (i.e., audio processing, video teleconferencing, full motion video). Real-world interfaces are best designed in an incremental process whereby progress is monitored by building and testing successive design revisions.

METHODOLOGY

Figure 1 shows the traditional design flow for a PCI-based function. Starting at the top in this figure, the design starts out as an architectural specification typically using an HDL. The core ASIC(s) are designed and then simulated as individual units. Based upon simulation results, several design iterations are typically required before the ASIC is ready for fabrication. The Printed Circuit Board is then designed and fabricated while the ASIC is being fabricated.

Unfortunately testing can not be performed until after the ASIC and board return from fabrication and are tested together. Often bugs in the ASIC and/or board are found during functional test, system verification, and then bus compliance test. Since the ASIC and board may have to be redesigned and fabricated, these bug fixes are costly in both schedule and NRE cost. Also the software drivers must be written, debugged, and integrated with hardware, and field tested prior to production tooling.

Figure 2 shows the design flow using System Emulation. Although the ASIC and board design steps are the same, inserting the System Emulation step actually streamlines the design process by eliminating ASIC and board design/fabrication iterations. Now the ASIC and board netlists are verified together as a system prior to starting the physical design. Also, system verification and bus compliance testing occur much earlier in the design process.

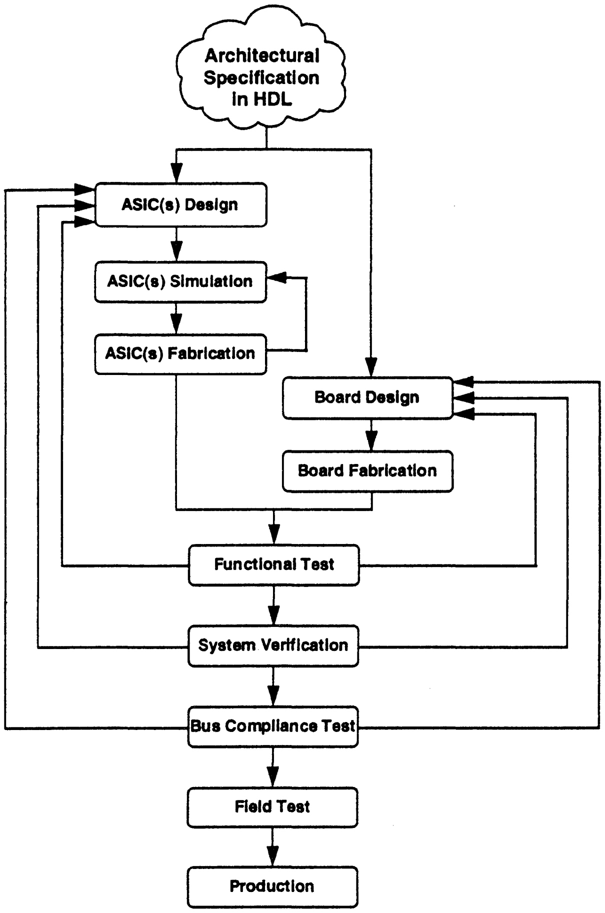


Figure 1 Traditional Design Flow

Moreover, the software drivers can be tested within the system emulation hardware—concurrent with the hardware design. Since the design has not yet been committed to silicon, it is even possible to explore software/hardware architecture tradeoffs. System emulation hardware can also be used for field testing with real customers.

Automated debug tools integrated into the system design isolate root problems straight away, rather than being diverted by contributing problems. Since the system is reprogrammable, various architectural specifications can be explored, the design can be performed in increments, and portions of the design can be re-used.

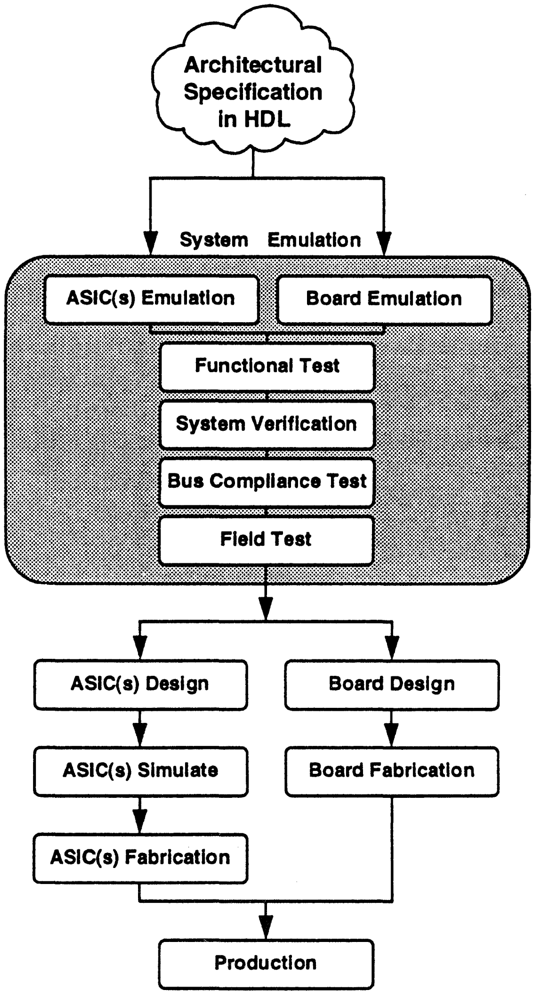


Figure 2 Design Flow with System Emulation

SYSTEM EMULATION PLATFORM

The system emulation platform is based upon Aptix proprietary interconnect technology. This interconnect technology consists of FPIC™ (Field-Programmable Interconnect Component) devices mounted on an FPCB™ (Field-Programmable Circuit Board) printed circuit board under the control of partition, place, and route algorithms within the development system software. Implementation is straightforward: user components are mounted on the FPCB, the design specification describing the netlist is imported into Aptix's development software, then the FPIC electrically connects the user components using routing information from the user's netlist.

The FPIC Device

The FPIC device (Guo et al, 1992) is a universal interconnect array with 936 user interconnect pins capable of connecting any pin to any other interconnect pin(s). This component employs a segmented routing array architecture that offers greater interconnect flexibility than traditional crossbar array architectures.

Figure 3 shows the internal structure of the FPIC device. The FPIC die has 1,024 I/O pads arranged in a 32 by 32 matrix. Associated with each I/O pad are I/O tracks that connect into the routing tracks that are grouped into horizontal and vertical routing channels. Routing tracks are segmented into various sizes to efficiently accommodate signal paths with different lengths. Shorter segments allow the same track to be shared by multiple signals when each runs only a short distance. Longer segments provide uninterrupted routing paths for signals running long distances.

The FPIC device is programmed by selectively writing to the SRAM cells to connect any pin on the FPIC to any other set of pins. Connections between routing tracks are made with bi-directional pass gates controlled by SRAM cells that contain the FPIC programming data. The configuration data is loaded at system power-up and may be re-loaded on command.

The FPIC device is available in two versions: FPIC/D and FPIC/R. The two components can be used together on the same board (the FPIC/R mounts on top of the FPCB while the FPIC/D mounts on the bottom). The FPIC/D offers a diagnostic capability in which 64 user diagnostic pins on the FPIC/D are connected to a logic analyzer through a flex cable. Since the FPIC device is connected to every user component on the circuit board, any net in the netlist can be probed by the logic analyzer under software control.

The FPIC/R is intended for stand-alone operation and does not have the diagnostic capability of the FPIC/D. In stand-alone operation, an SPM (an EPROM-based Stand-alone Programming Module) configures the FPIC/R on power-up so the board can be used without being connected to a host computer.

The FPCB Printed Circuit Board

The Aptix FPCB is a Field Programmable Circuit Board that is used with the FPIC device. The FPCB architecture is shown in Figure 3. The FPCB contains a number of through-holes that can accept user component pins. Each through-hole is connected to a separate pin on the FPIC device. Connections between user components are made by routing the FPIC device to implement the user's netlist.

The FPCB is divided into regions that are each supported by a separate FPIC. Routing between components in different FPIC regions is implemented with board-level interconnect that directly connects two FPIC devices. For example, the three components shown in Figure 3 are connected with two FPIC devices and global interconnect between the devices.

Several types of FPCBs are available, each optimized for a specific application. FPCB optimization involves: different global interconnect architectures and component capacities; the ability to mate with adapters to implement standard bus protocols; and board-level structures to implement I/O interfaces, low-skew clocks, and system busses. Different types of FPCBs include:

- General Purpose (GP) type available in the VME 6UE and PC/AT form factors
- ASIC Prototyping (AP) type available for ASIC emulation applications
- Multi-Purpose (MP) type for system emulation applications

The GP4 and MP3 are discussed in this paper due to their respective suitability for PCI motherboard and accessory card applications.

The GP4 FPCB

The GP4 FPCB is uses the industry-standard Type C hole pattern whereby the through-hole pattern is arranged with repeated rows that are spaced at alternating 100 mils and 300 mils apart. This pattern provides high density (50 pins/sq. inch) and allows DIP (both 300 mil and

600 mil) and SIP components, surface mount components (such as QFP and PLCC), and PGAs to plug in. Discrete components (such as resistors, capacitors, and diodes) can be mounted using headers or daughter boards. A rich set of adapters is available to mount standard connectors, including:

- PCI Local Bus expansion slot connector
- PC AT expansion slot connector
- PCMCIA I/O connector
- 72-pin SIMM memory module connector

The GP architecture provides for system-level considerations such as low-skew global pins for clock drivers and power distribution through readily accessible power and ground lands. The GP4 board supports 3,000 component pins (or about 100 components).

Because of its ability to mount user components, the GP4 is suitable for board emulation—such as PCI motherboard applications (Long, 1994).

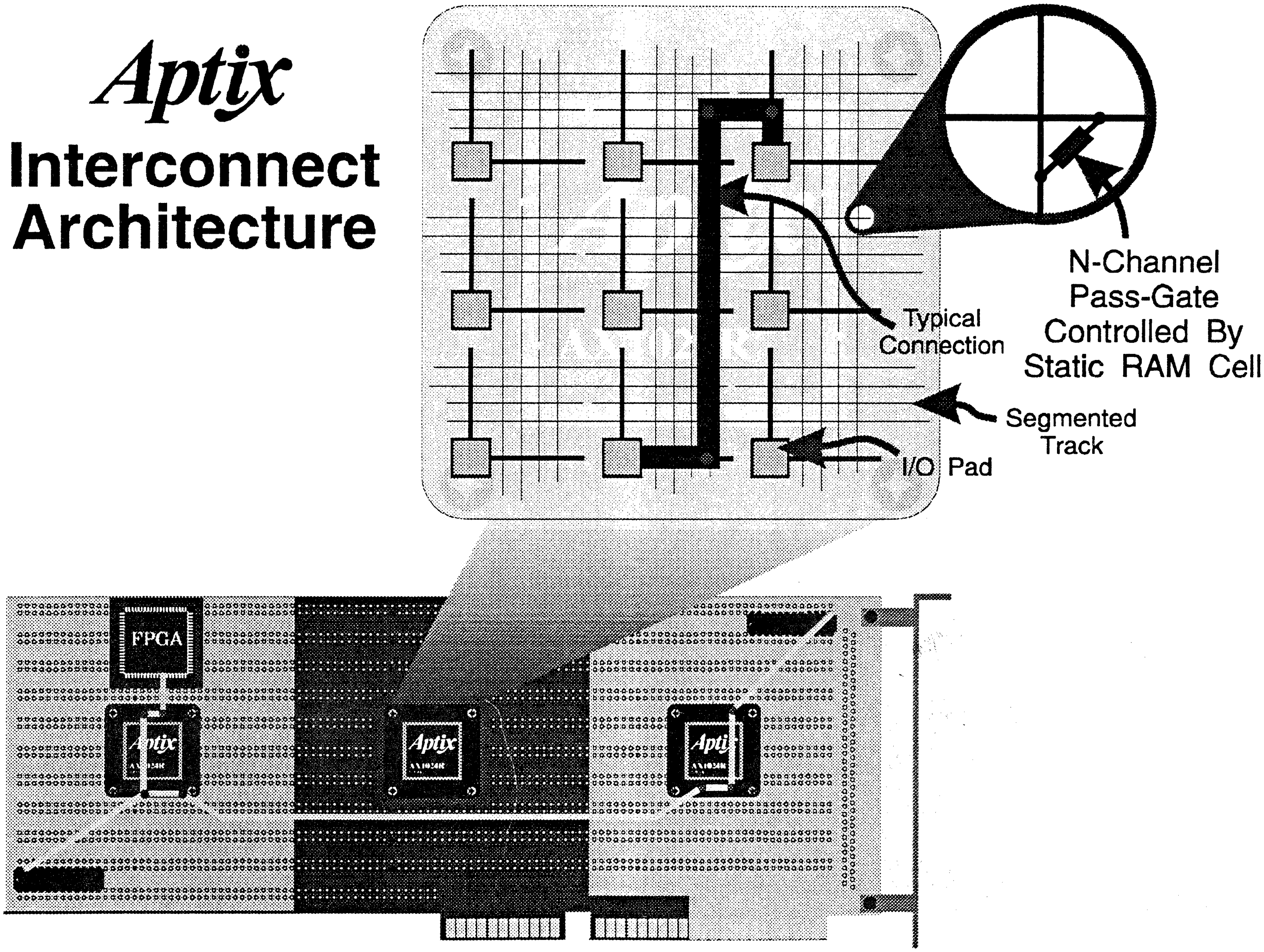


Figure 3. Interconnect Architecture of FPIC™ and FPCB™

The MP3 FPCB

The MP3 enables system emulation by providing the capability to emulate a board (using GP4-style through-hole regions) and an ASIC (using reprogrammable FPGAs) together.

The MP3 through-hole region is also implemented as a standard Type C pattern. Therefore user components can be mounted directly on the MP3 for board emulation.

A row of special-purpose pins along the edge of the through-hole region provide power and ground, access to low-skew clocks, and access to FPGA configuration pins. FPGA-specific adapters are keyed to access these signals. Therefore reprogrammable FPGAs are conveniently mounted on the MP3 for ASIC emulation.

The MP3 also provides high-speed busses that can connect to FPGAs or components in the through-hole region. These busses are useful for system emulation tasks where part of the design must run over 50 MHz.

The MP3 is shown in Figure 4. In the MP3 FPCB, the general purpose component hole area is divided into three FPIC regions that support a total of 1,900 component pins (or about 60 components). This FPCB will also support 12 FPGAs mounted on adapters and 2 FPGAs mounted in the I/O socket or a combination of adapters and components.

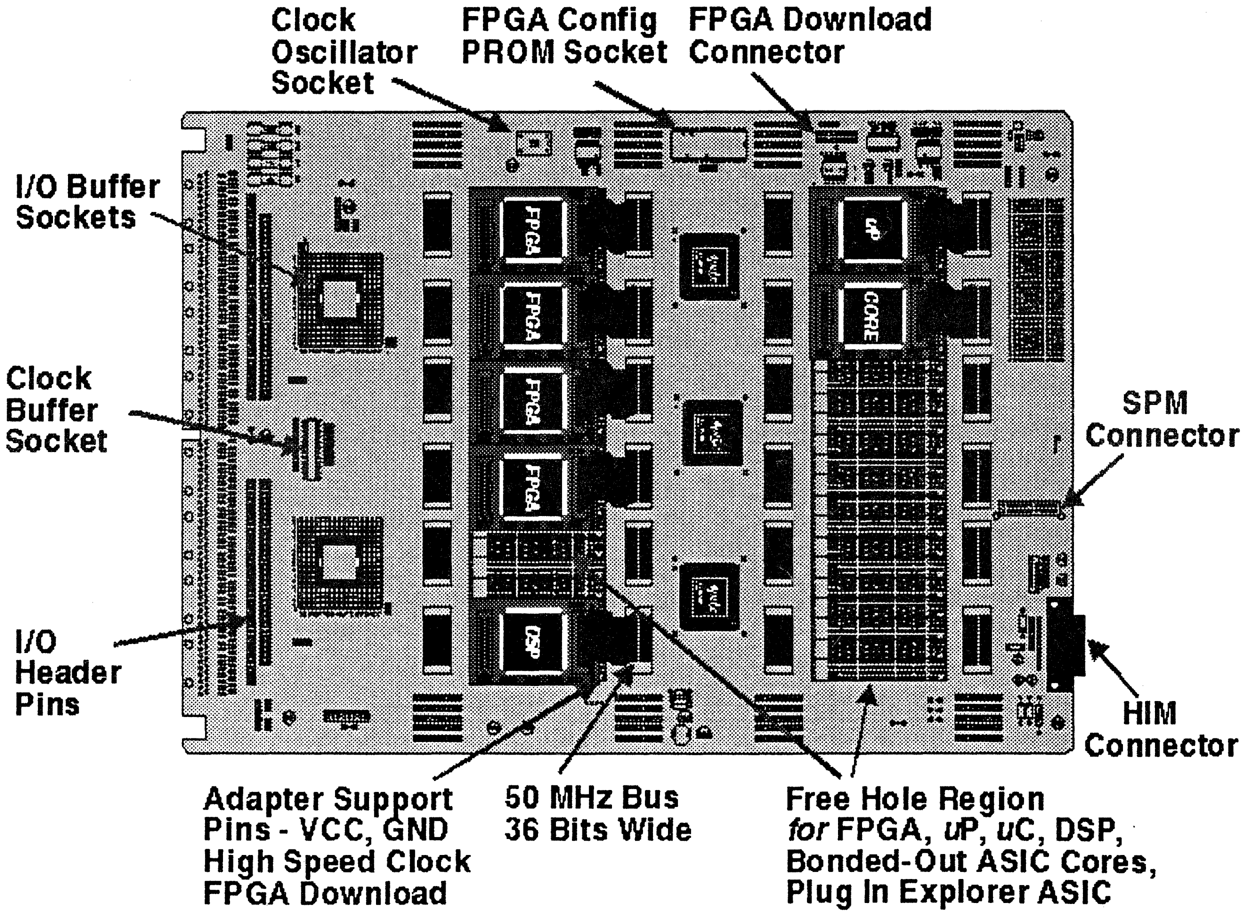


Figure 4. MP3 System Explorer FPCB

SUMMARY

Today's PCI development can be quickly implemented and verified using system-level emulation tools. Starting from the HDL or schematic entry, the PCI design is automatically downloaded to reprogrammable hardware, interfaced to other components and powered up for full system integration and test.

System level emulation and rapid prototyping are made possible by: (i) software which automatically maps synthesized Verilog and VHDL to FPGAs and converts board-level schematics to a netlist; (ii) high-performance hardware including FPICs and FPCBs. The typical turnaround time from PCI simulation to hardware is less than three hours. Once implemented, test and debug operations are under software control which allow any pin or net to be observed on a logic analyzer quickly.

REFERENCES

R. Guo, H. Nguyen, H. Verheyen, H. Cai, H. F. S. Law, A. Mohsen. 1992. "A 1024-pin Universal Interconnect Array with Routing Architecture", Proc. IEEE Custom Integrated Circuits Conference. pp. 4.5.1-4.5.4, 1992.

Courtney Long. 1994. "Functional Verification of Microprocessor-Based Systems Using PLD Technology", PLD Design Conference. pp. 1.1.1.A, April 11-13, 1994.

System Data Book, Aptix Corporation, San Jose, CA, 1993.

BOUNDARY-SCAN IN PCI BUS SYSTEMS

Menachem Blasberg
Corelis Inc.
12607 Hidden Creek Way
Cerritos, CA 90703

ABSTRACT

Since its introduction as an industry standard in 1990, Boundary-Scan - also known as JTAG - has enjoyed growing popularity for board level manufacturing test applications. Due to its low cost nature and IC level access capabilities of the boundary-scan standard, its use has expanded beyond traditional board test applications into design and service. New industry standard buses such as the PCI bus, and soon, the VME bus, are adopting boundary-scan as the common test bus interface. This presents new possibilities for the use of boundary-scan at the system level rather than just the board level. System level test is supported using a hierarchical scan architecture such as the SCANPSC110 bridge concept developed by National Semiconductor. This article provides a brief introduction to boundary-scan and its application to the PCI bus. Hardware and software development and test tools that support hierarchical scan are also reviewed.

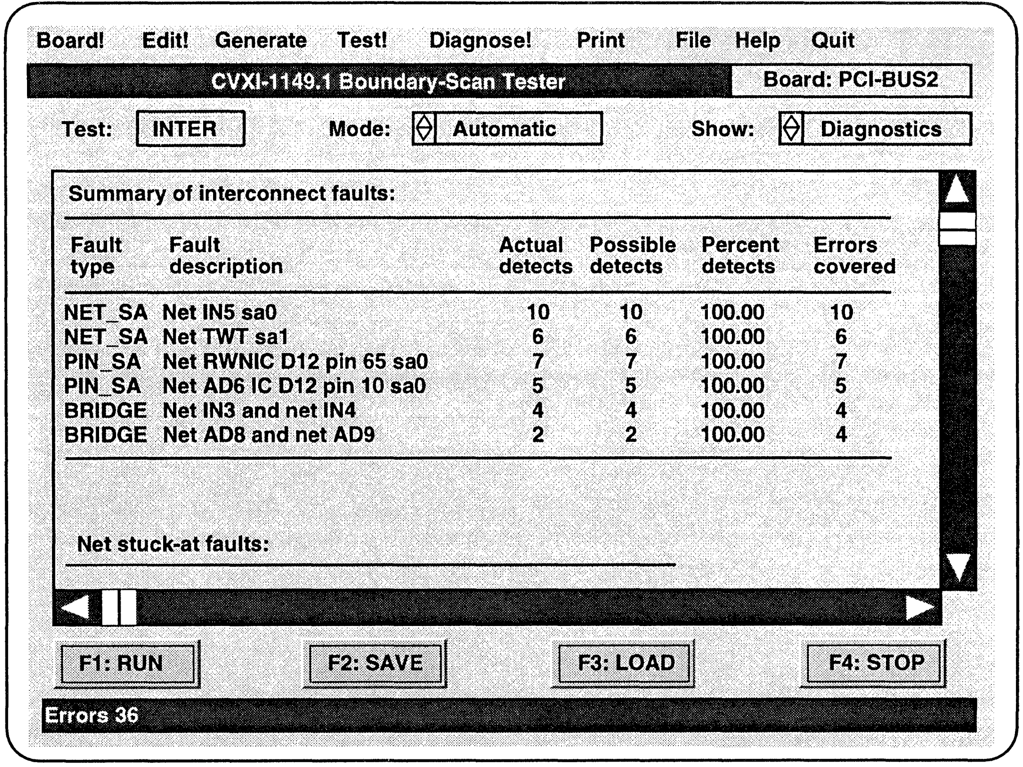
Introduction to Boundary Scan

As electronic systems continue to increase in complexity and capability, and as circuits and packaging densities continue to rise, system-level development and manufacturing techniques face ever-higher hurdles. In 1990, the IEEE 1149.1 "boundary-scan" test standard was created to address the problem of how to test modern digital systems. This standard was originally created to ease the testing of dense boards in a manufacturing environment. Boundary-scan has since become a commonplace companion to traditional in-circuit test techniques. Most of today's manufacturing testers offer boundary-scan capabilities, and "design for test" (DFT) features such as scan are gradually being embraced by designers who face daunting test issues.

To enable boundary-scan-based testing, digital ICs on the board under test must be furnished with scan circuitry at each I/O pin. These scan I/O cells are all connected in a serial shift register enabling the logic state of each I/O pin to be observed and/or modified by shifting data patterns in and out through the serial scan chain. This scan test logic is only enabled during a special test mode, allowing the system to perform its normal "mission" operations at all other times.

The observability and controllability-enhancing capabilities of boundary-scan have been used to good advantage in board testing, where they provide access to what might otherwise be unreachable "buried" nets - nets for which no vias or test points are available. This is especially valuable in systems that use fine-pitch high lead-count packages that are surface-mounted on double-sided boards, which also lack through-holes. Such boards can not be tested using traditional "bed of nails" contact fixtures.

By allowing direct access to nets, boundary-scan eliminates the need for the large number of test vectors which are normally needed to properly initialize sequential logic. A few tens or hundreds of vectors may do the job that had previously required thousands of vectors. Potential benefits realized from the use of boundary-scan are shorter test times, higher coverage, increased diagnostic capability and lower capital equipment cost. The screen image below (Figure 1) illustrates the high level of test coverage and detail of diagnostic information available through boundary-scan testing. This picture was taken from a Corelis and JTAG Technologies CVXI-1149.1 Boundary-scan tester for VXI based test systems.



Board: PCI-BUS2

Test: INTER Mode: Automatic Show: Diagnostics

Summary of interconnect faults:

Fault type	Fault description	Actual detects	Possible detects	Percent detects	Errors covered
NET_SA	Net IN5 sa0	10	10	100.00	10
NET_SA	Net TWT sa1	6	6	100.00	6
PIN_SA	Net RWNIC D12 pin 65 sa0	7	7	100.00	7
PIN_SA	Net AD6 IC D12 pin 10 sa0	5	5	100.00	5
BRIDGE	Net IN3 and net IN4	4	4	100.00	4
BRIDGE	Net AD8 and net AD9	2	2	100.00	4

Net stuck-at faults:

Errors 36

Figure 1 : Diagnostic Screen of PCI Motherboard Boundary Scan test

Recently, ways have been found to extend boundary-scan testing to the level of entire systems. This allows a single test access mechanism - boundary-scan - to be used at different hierarchical levels of a system - components, boards, modules, sub-systems, etc. Test-patterns for one level can be reused at a higher level; for example, board test-vectors can be applied to a board by a system-level test master after that board has been built into a system, allowing systems to be assembled and tested incrementally.

Standard Buses

Standard system buses, such as PCI, are important primarily because they allow complete systems to be easily assembled from compatible sets of "off the shelf" boards. Standard buses provide a uniform interface that allows a conforming board to be used in multiple systems produced by multiple vendors.

It is possible to use boundary-scan in conjunction with standard buses. If a bus standard makes provisions for scan usage as part of the bus standard, then that standard defines a uniform test-access mechanism that can be supported by any standard-compliant boards. This mechanism can then be used both to test individual boards, and to test complete systems consisting of boards and their interconnect (backplane and cables. The PCI bus standard has adopted boundary-scan as an integral part of the PCI bus.

The PCI bus is widely finding favor as a high-performance, block-transfer-oriented internal system bus, primarily in personal computers and workstations. Often, these are very high volume products, for which manufacturability and field maintainability are key economic issues. In addition, intense market pressures force ever-shortening development cycles for such products, making ease of development a key pre-requisite. Each of these three constraints can benefit from the use of boundary-scan as a system-level testing and debug vehicle.

Extending Boundary-Scan to the System Level

As mentioned earlier, boundary-scan was originally developed for testing boards in a manufacturing environment. Applying these techniques at the system level requires extension of the IEEE 1149.1 standard. In the present context, a system consists of a set of boards installed in slots on one or more backplanes. While it is possible to extend a single system-level scan chain through the scan-compatible components on all boards, there are disadvantages to this. First, a fault in any component's scan circuitry disrupts scan access to all chips. Second, this approach requires the use of very long test vectors which tends to increase test times. Figure 2 illustrates this configuration.

To overcome these drawbacks, it is necessary to use multiple scan chains to access various portions of the system. This can be done by running multiple scan chains from the test master (say, one per board in the system), and then using path-multiplexing circuitry to allow the master to selectively access different portions of the system. This has the disadvantage of requiring the use of a complex backplane.

A more elegant and less costly approach is to use scan addressable scan chain selection circuitry in order to partition the system's scan network. That is, special "gateway" circuits can be addressed by the test master to provide access-on-demand to the various branches of the boundary-scan network. This is the approach employed in National Semiconductor's SCANPSC110 bridge architecture. With such a mechanism, the system test master is able to selectively apply test patterns to specific boards within a system, or even to sub portions of a given board (down to the level of individual ICs, if desired).

Since boundary-scan tests can be implemented completely in software, test vectors already developed and used during production test can be run on the embedded processor of the product that is shipped. As such, a central processor can access any card inserted into the PCI backplane for diagnostic and auto configuration purposes. This level of built-in test surpasses most conventional methods used in the past at virtually no incremental cost. Since system configurations change over time, the multi-drop (i.e. parallel-gateway) capability of the scan architecture is particularly suited for these applications. Figure 3 illustrates a PCI based system using a multi-drop boundary-scan bus to provide built-in test of card modules.

Scan on Standard Buses

There is a natural synergy between the use of standard buses such as PCI in building a system, and the use of boundary-scan as a means for testing such systems. Bus standardization and test-mechanism standardization are both methods for allowing systems to be constructed in a modular manner. This offers advantages in terms of scale of manufacture, and in terms of ease and versatility in configuring application-specific systems. The

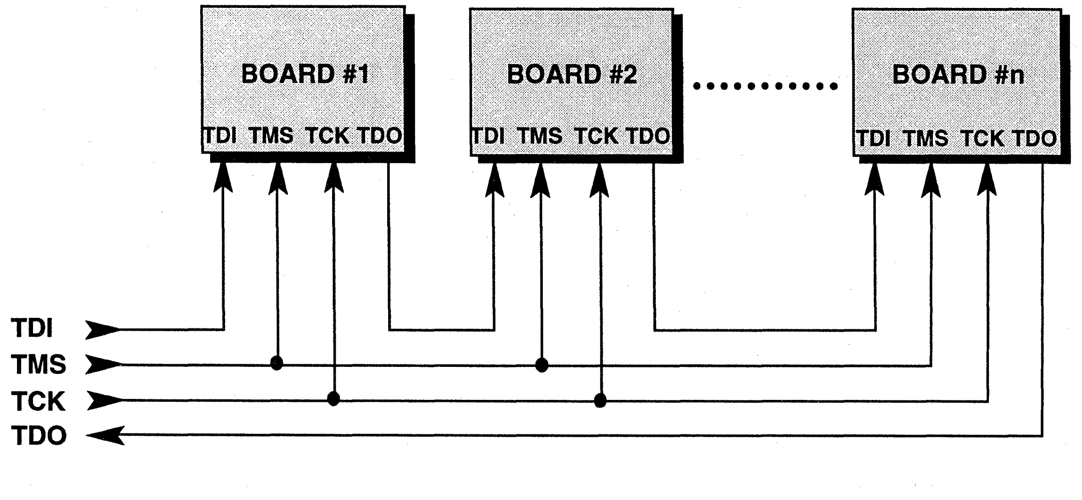


Figure 2: Typical "Single-Chain" Backplane

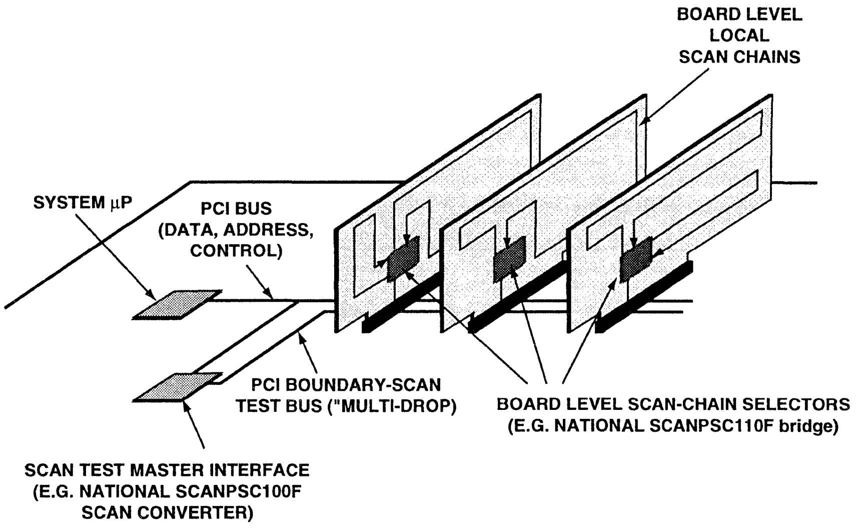


Figure 3: PCI based system using a multi-drop boundary scan bus

following sections describe more in detail how these advantages apply to PCI-based systems.

PCI-Bus Standard Boundary-scan Access

The PCI bus standard provides the necessary boundary-scan signals to support scan test. The relevant pins on the PCI bus connectors are shown in Table 1.

Implementing System Level Scan on the PCI bus

Fortunately for those contemplating taking advantage of boundary-scan technology for reasons stated previously, the necessary boundary-scan development and test tools to support PCI system level hierarchical scan exist today. This is consistent with the concept of the IEEE-1149.1 boundary-scan standard to provide a product life cycle test philosophy. The same boundary-scan resources can be used in design, manufacturing and service with no additional investment required in test software development.

In product development, boundary-scan can be used to enhance an engineer's observability and controllability of the system. For example, many modern microprocessors are equipped with a scan-

Pin	Side B	Side A	Type	Description
1		TRST#	input	Test Reset Not
2	TCK		input	Test Clock
3		TMS	input	Test Mode Select
4	TDO	TDI	output/input	Test Data Out / Test Data In If boundary scan is not supported by the PCI plug-in card, these two pins need to be shorted together.

Table 1: Boundary-Scan pins on the PCI bus connectors

based emulation port that provides access to internal processor resources such as registers. These resources can be interrogated and modified, code can be downloaded and single-stepped and breakpoints can be set through a common four wire JTAG interface, thus lowering the cost of development tools such as in-circuit emulators. In a multi-board backplane PCI-based system, the same JTAG-based emulator controller can be used to access any board on the backplane using the hierarchical addressing capabilities of a system level JTAG implementation. JTAG based processor emulators are available from companies such as Corelis Inc., that specializes in scan based development and test tools.

In order for boundary-scan to be useful as a generic test vehicle throughout the product life cycle, this test technology must be supported by appropriate hardware and software tools. On the hardware front, tester hardware must be available to drive the scan chain(s). In a standard-bus environment such as PCI, this hardware must mesh properly with the standard bus. In terms of software support, users must be able to generate test vectors in as automated and trouble-free a manner as possible. In addition, it is very useful to be able to use scan as a vehicle to observe and control the operation of all portions of the system. This can be achieved by using the boundary-scan emulation port that is usually available on the central resource of the system: the system microprocessor.

The suite of test tools mentioned above must be available in a variety of application environments. Certain tools, such as scan-based processor emulation control, are needed primarily during product development. Other tools, such as high-throughput scan-compatible testers, are needed to support large-scale manufacturing. Still other tools, such as a compact boundary-scan tester, are useful to service and maintain products once they are in the field. Boundary-scan can serve as the "common denominator" across all of these tools and application environments.

Boundary-Scan Test Tools

While Boundary-scan Testers are now available from several vendors, only a few, such as Corelis and JTAG Technologies, support hierarchical scan architectures. Some of Corelis and JTAG Technologies' test controllers such as the PC-based Explorer, VXI-bus based CVXI-1149.1 or the high speed IEEE-488 bus compatible VectorBlaster are capable of supporting high test throughputs and complex production test requirements with software support available for multi-drop scan hierarchies. These products not only support go no-go testing but offer in-depth node-level diagnostics as well, thus greatly reducing rework cost and board fall-out. These types of testers are particularly well suited for high volume production environments such as those found in the manufacture of PCI based PC's.

The same JTAG access can be utilized for hardware debugging of PCI-based designs such as PC motherboards using the PCI pre-processor available from Corelis Inc. This logic analyzer pre-processor supports Hewlett-Packard's popular line of logic analyzers. Using this pre-processor not only provides the design engineer with state and timing information on all PCI signals, it also contains a JTAG trace module that allows the logic analyzer to disassemble all boundary-scan chain TAP controller states. All scan controller TAP states can be captured and analyzed using this pre-processor. A standard JTAG test connector is provided on the pre-processor to enable the user to connect any boundary-scan tester for external access to the PCI bus boundary-scan chain. This enables the backplane as well as any board connected to the PCI bus to be accessed in case a JTAG connector was not provided on the backplane itself. This scenario is shown in Figure 4.

Automatic Test Vector Generation for Hierarchical Scan Designs

There are currently several vendors who offer Automatic Test Pattern (ATPG) Generation Software for board level boundary-scan implementations. One is the Boundary-scan Test Pattern Generation software package (BTPG) developed by JTAG Technologies B.V. — a spin-off from Philips International of The Netherlands — and available in the US through Corelis Inc. The BTPG package is based on a high level structural test description language known as BTSL (Boundary-scan Test Specification Language). This hardware tester independent language includes support for hierarchical scan designs such as those available from National Semiconductor and Texas Instruments.

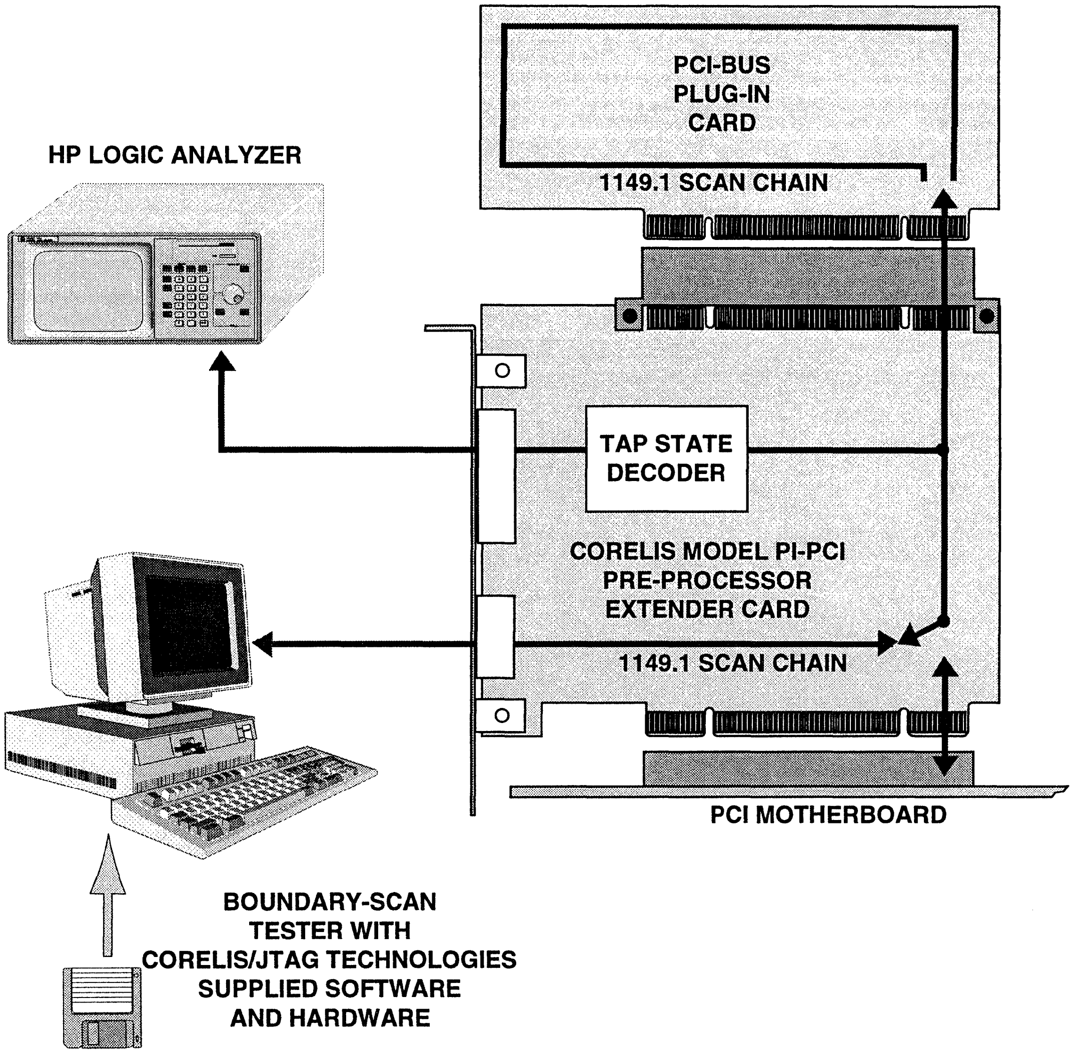

HP LOGIC ANALYZER
PCI-BUS PLUG-IN CARD
1149.1 SCAN CHAIN
TAP STATE DECODER
CORELIS MODEL PI-PCI PRE-PROCESSOR EXTENDER CARD
1149.1 SCAN CHAIN
PCI MOTHERBOARD
BOUNDARY-SCAN TESTER WITH CORELIS/JTAG TECHNOLOGIES SUPPLIED SOFTWARE AND HARDWARE

Figure 4: Boundary-Scan Integration with Corelis' PCI Preprocessor

To generate test vectors for mutli-drop scan chains, the BTSL language supports a scan chain interface logic description syntax that allows the architecture of the scan chain interface to be described. This is best illustrated using a simple example. Given the National SCANPSC110F bridge-based multi-drop backplane design shown in Figure 5, the following BTSL interface logic description would apply:

```
SYNTAX_VERSION          1.2
DESIGN                  PCIBCKPL
REVISION                1C
TESTER_CHANNEL TAP1
    MULTIDROP1      (PSC110F, 1, TAP1, TAP2, TAP3)
    MULTIDROP2      (PSC110F, 2, NONE, TAP4, TAP5)
    MULTIDROP3      (PSC110F, 3, TAP6, TAP7, CAS-
                    CADE1)
    CASCADE1        (PSC110F, 4, TAP8, TAP9, NONE)
END_CHANNEL
```

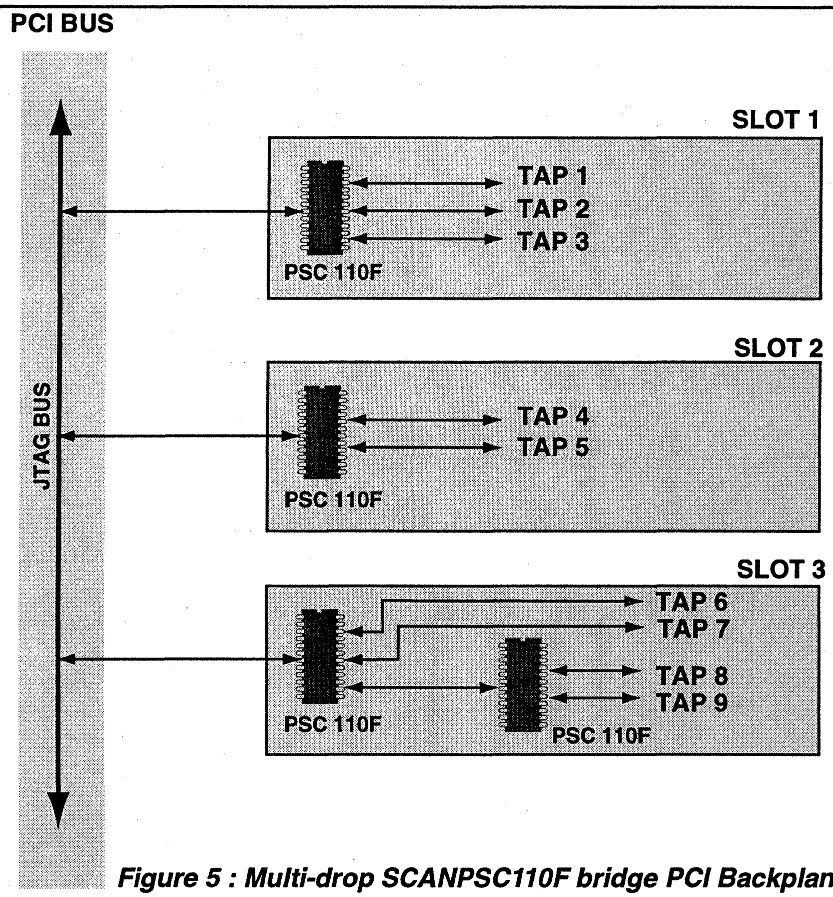


Figure 5 : Multi-drop SCANPSC110F bridge PCI Backplane

From this simple description, the BTPG software is capable of generating test vectors that fully test all interconnects on each individual plug-in board and provides node-level diagnostics. Any required chain addressing and switching is handled automatically by BTPG. The level of diagnostics can be selected depending on the test execution platform to be used and the desired test execution times.

CONCLUSION

Boundary-scan offers much promise as a generic system-level test mechanism that is usable throughout the life of a product. Scan techniques are especially powerful when implemented in standard-bus-based systems, as a natural extension of the backplane standardization for those systems. The PCI standard is a prime example of a standard bus that support this synergy. In such a standardized environment, silicon manufacturers, test-tool vendors and board and system designers can more productively work together to produce well-integrated solutions to system-test problems.

A DESIGN FOR REUSE (DFR) METHODOLOGY FOR THE RAPID DEVELOPMENT AND VERIFICATION OF PCI PROTOTYPE AND PRODUCTION SYSTEMS

Alak Deb, J. Scott Runner, Maulin Bhatt
Synopsys, Inc., DesignWare™ Components R&D
700 East Middlefield Road, Bldg. B
Mountain View, CA 94043-4033

ABSTRACT

Design of PCI-based systems poses not only technical challenges, but the designers of these systems often work under demanding time-to-market constraints. During design, opportunities to explore the space of different implementations and configurations often yield to schedule pressures. Moreover, derivative systems of original designs that may impose new requirements are often developed without effective utilization of previous PCI designs, nor embodying enhancements derived from personal, industry and field experience. As a consequence, even second generation PCI systems often fall short of the overall system performance improvements that they could leverage, and take more time to design than required. In this paper, a design methodology using the DesignWare™ PCI controller MacroSet will be presented which provides for fast implementation, verification and successive refinement based on feature, performance and environment changes, and evolution of the PCI specification. Several PCI design considerations and how they translate into implementation will be discussed regarding what optimizations and improvements they introduce. The methodology will involve the implementation of a real-time multi-media add-on card in a FPGA prototype and production ASIC, the tools used to validate and optimize the system, and several design tradeoffs and their cost and effect. Note that this paper will focus on design of "add-in" cards, however most of the concepts are relevant to bridge designs, though there are additional issues that must be considered therein.

THE DESIGN FLOW

A High-level Design (HLD) methodology is employed leveraging high-level synthesis of VHDL or Verilog. HLD does not directly equate to "Top-down" design, but does keep the designer abstracted from as much irrelevant detail as possible, enabling the designer to focus on their value-added, and to more quickly implement their system.

This design flow (depicted in Figure 1) assumes top-down design of the system, with modular construction of the PCI controller by leveraging the existing DesignWare™ MacroSet. Test development is by first testing the PCI-controller under the required configuration, testing the application separately, and then integrating them. Bottom-up or top-down test approaches may also be employed, depending on one's existing, well-defined and effective methodology. Test development is discussed later, but an important point to make is that tests should be straightforward to develop, and easily reusable. For example, as the design is successively refined, tests from higher levels of abstraction (e.g. behavioral/RTL should be derived for unit-level and pre and post-layout gate-level simulations). Furthermore, with tools such as Synopsys VHDLGen product, sub-system and unit-level tests and test benches can be derived from higher integration tests.

The design flow referenced in this paper relies on the top-down approach, but one in which the PCI controller is implemented by modifying Synopsys' existing MacroSet implementation by removing modules, if needed, and by specifying required parameters. Some modifications to the provided test bench and test suites may be required. This sub-system test bench and test can now be simulated to verify compliance and correct functional operation (Figure 1). The controller portion of the design is now done.

In parallel, the application is being architected, partitioned coded, simulated and synthesized. Remember that during this process, it's important to perform synthesis early in the process to insure efficient partitioning and coding of the design, as well as testability. Once a synthesizable version of the application exists, or at least the interface between the PCI controller and the application, synthesis constraints and scripts can begin to be modified. Sets of these constraints and scripts for the MacroSet are provided, but require modification based on changes to the MacroSet and the application interface. False paths and Multi-cycle paths within the MacroSet are already provided. Top-level and application constraints, including multi-cycle paths and false paths must be identified, and then synthesis and timing analysis can begin.

To effectively support technology independence, instantiation of technology-specific gates should be eliminated or kept to a minimum. Types of cells which typically need to be instantiated are I/O pads and large muxes, RAMs and ROMs. I/O pads should be partitioned into their own level of hierarchy, if RTL descriptions of these pads are also defined for this partition, then the insert_io_pads function can be leveraged when mapping to FPGAs and those ASIC libraries which support I/O pad synthesis. By minimizing instantiation of technology-specific cells, and by partitioning the design into modules which can be contained within individual FPGAs, the design can be targeted in a straightforward manner into FPGAs, as well as alternative ASICs (Xilinx, 1994).

After functional simulation and synthesis are complete, unit delay gate-level simulation was performed. This is to insure that the circuit is correctly synthesized, that timing loops, if they exist, are legitimate, and that the integrated design functions properly, without interference of numerous timing violations which undoubtedly will exist after the first pass(es) of synthesis. The initial synthesis runs also need not have detailed timing information. The concept is to begin by pruning out gross errors, then successively refining the circuit until all functional and timing violations are removed. Once unit delay is clean, then (or in parallel) false paths should be eliminated and multi-cycle paths identified, while applying top-level timing constraints. Details regarding alternative synthesis compile and constraining methodologies are beyond the scope of this paper. However, this specific design flow involved using the compile-characterize-revise-write_script methodology, while allows for accurate timing constraints, while facilitating automation of synthesis scripts.

When static timing demonstrates that all timing errors are removed, then the design can proceed to layout. PCI imposes demanding timing requirements, such as setup and hold times, and output delays into a 50pF load (t_{VAL}). To optimize these timings requires involvement during floorplanning and/or place & route, depending on the technology employed. The DesignWare™ is entirely register isolated at the boundary of the PCI interface (super-synchronous). Therefore, most of the critical timings can be addressed by abutting the MacroSet interface registers adjacent to their associated I/O pads.

Post-layout static timing analysis and simulation is highly recommended, as pre-layout estimates may be overly conservative or liberal but either way, result in variations that are nearly always greater than the timing margins available. Post-layout simulation is generally required by ASIC foundries for sign-off, so use it to one's advantage. This is accomplished by back annotating the delay, typically in a Standard Delay Format (SDF) file, as shown Figure 1. ASIC vendors generally also require functional manufacturing test patterns which comply with certain tester requirements. Since these timing requirements generally do not map well to the system timing under which the functional tests are derived, tests must generally be derived for this purpose. The VHDLGen product captures stimulus and response at the boundary of the ASIC and can "till" the timing of transitions to match characteristics specified by the designer.

Note that if Test Compiler is used to insert full or partial scan chains into the design and execute Automatic Test Pattern Generation (ATPG), then these tests must be converted into the appropriate format.

DESIGN REUSE AND EXPLORATION

The goals of HLD are to 1) greatly increase productivity by abstracting the designer to a level wherein more logic and functionality can be expressed much more succinctly (i.e. Hardware Description Languages, technology independence, and high-speed simulation), and 2) to facilitate Design Reuse. Technology migration of a design is but one facet of reuse. Others include leveraging existing designs for subsequent product families and derivative products. Design Reuse is also the process of how efficiently the design can be reused throughout phases of the design itself. For example, functional and timing errors require design iterations, as does the exploration of different characteristics. The cycle time of these analyze-modify-verify loops is critical to implementing an efficient design in a short amount of time. Shortening this loop means faster time to market, also affords the ability to explore alternate architectures and to rapidly derive new systems and upgrades.

The approach taken by Synopsys DesignWare™ is to support design modifications through adjustment of parameters and design constraints. At the inception of the project, then designer would evaluate the functional requirements of the system, and make a pass at the required characteristics of the controller. There are three ways in which changes to functionality and characteristics can be affected: assembling MacroSet modules in different configurations, modification of parameters, and modification of synthesis constraints. Figure 2 depicts a slave-only implementation with the MacroSet, while Figure 3 illustrates a master-slave configuration. Multiple datapath configurations are also possible. Design parameters such as configuration space characteristics, interface and handshaking with the application, datapath construction (number of data/address paths, whether register or FIFO, and if FIFO of what depth), are all important design decisions that may require revisiting as more design data and system behavior becomes available, or when given designs are to be modified for subsequent use in a next generation card. Since these parameters may change frequently, they have been implemented as DesignWare™ parameters for rapid implementation and modification.

Reusable components, in which each component is a black-box, also affords the ability to accommodate new optimizations of the PCI controller, as well as new specification versions. And it provides for alternate constructions of given modules. For example, versions of a module can be area-optimized, performance optimized, optimized for testability or power, and yet support the same interface to allow for plug-and-play. Thus, a reusable design becomes valuable in that it can be more easily learned and modified by designers taking over an existing project for future upgrades or new architectures. Now that mechanisms for supporting Design Reuse have been reviewed, the system-level considerations that they support will now be discussed.

DESIGN TRADEOFFS

Architecting and designing PCI systems involve first evaluating design requirements and understanding the effect of various design parameters on add-in card and overall system performance, cost, robustness/quality and functionality. Designers know this process well, and the fact that these forces are often in competition with one another is a continual challenge. PCI tradeoffs include optional functionality which does not compromise compliance. Designers of "closed" or proprietary systems, for which PCI is just as equally a good choice, may be tempted to compromise compliance, however, this is taking risk since many requirements are thought out for the purpose of general, efficient interoperability, and doing so will demand more re-design effort should future revisions of such systems begin to open up to add-in cards or leverage standard platform systems/subsystems. Some of the design decisions PCI designers will face are included in the following sub-sections.

Bus Mastering

A minimal PCI controller implementation is shown in Figure 2. At a minimum, a PCI implementation must support a slave finite state machine (DWpci_slave_fsm) controller with address space recognition/decoding (DWpci_slave_arm) , a configuration space (DWpci_config), and an interface which electrically connects to the PCI bus (DWpci_interface). Bus Mastering is not required functionality.

The decision to use bus masters on add-in cards is based on the bandwidth requirements and characteristics, and source and destination of the data that will be transferred to/from the card. If data is primarily be pumped into the card from system memory/processor, or other add-in cards, such as in the case of graphics adapters, then slave writes are the primary source of bandwidth, with slave reads being used for state/status. If S/W or other H/W are to operate directly on images from a shared frame buffer on the card, then slave reads can be relied on to extract the data. Alternatively, and access via S/W to the data could be translated by the driver into master write from the card into the desired location. This can be accomplished if the master has DMA associated with it that accepts a starting address and transfer size, and a destination address. The transfer would need to be broken into lines in this specific case, unless a more sophisticated "2-D" DMA scheme was employed in which an entire window, etc. could be specified. In general, mastering is attractive if the data resides on the master's side, since data availability can then be tightly coordinated with the master state machine bus transactions.

Configuration Space

The heart of PCI's "DIP-less", reconfigurable architecture which supports Plug-and-Play (PnP) is the configuration space. The three parts making up this space are the required header, the optional header, and the device-specific registers. The required header must be programmed to include the identification/personalization information such as Vendor/device ID, class codes, etc. Also required are the status and command registers which communicate state information back to software in the former case, and allow for device configuration in the latter case. The required header also includes the Base Address Registers (BARs) which were discussed in association with address range allocation, and the Expansion ROM BAR. The requirement for implementing bits in the address space is often based on capability. If an agent does not implement certain functionality, then bits associated with that functionality may generally be tied low (inactive).

Other design decisions relative to the configuration space are whether to hardcode all bits, or make them configurable. While some bits are specified as read/write (R/W), others are read-only (RO). However, in many occasions, a PCI controller may be reused in different contexts, a scenario known as "common silicon". However, to do so requires that certain RO fields be programmable prior to POST time. This may be accomplished by implementing these RO bits as registers. At power-on-reset (POR), the state of these bits is programmed by a sequencer that retrieves the state to be programmed from external EEPROM or other source. While the configuration space must respond to all configuration reads and writes directed to its space, in the case that the configuration space is being loaded from EEPROM, it is acceptable to issue a disconnect-retry until the state has been programmed. Therefore, the ability for the slave FSM to determine when the config space is ready is important.

Another issue when dealing with the configuration space is that of the device-specific registers. One should not be immediately inclined to move all control and status registers in their card or system into config space; recall that accesses to this space require at least 2 cycles to complete. Therefore, it is generally recommended that card status, command and control registers and memory spaces such as FIFOs be allocated to memory space, or memory and I/O spaces.

Application-side Interface

How the PCI controller interfaces with the application is critical to overall system operation. Often the application is running at a different clock domain than the PCI domain. In such a case one has to be careful to synchronize the asynchronous signals at the interface (such as control lines and flags). While this introduces a 2 clock latency (assuming that a dual stage synchronizer is used) in the signals, it insures that signals are de-glitched and that the probability of meta-stability is radically reduced. Look-ahead decoding can help mask the synchronization latency. For example, FIFO full and empty flags can instead be defined as "low-water" and "high-water" marks which are either a line, or a couple of DWORDs or more from the top or bottom, depending on the synchronization latency, and the application response latency.

Another factor that influences how the application interface is designed is the degree of visibility that is required of the PCI interface. Generally, the interface between the and the application entails data/address and control and status signals. A loosely-coupled system in which the application supplies data, start address, and number of bytes/DWORDs to transfer to a DMA controller and then lets the transaction, including exceptions, be managed by the controller is one solution. An alternate solution is to allow the application to have visibility into the state of the controller, and be directly involved in exception handling. This allows the application more control, which is sometimes useful when the application is arbitrating among multiple resources.

Datapath Architecture

This is perhaps the biggest issue in efficient PCI design. Latency and Datapath Buffering Latency has always been a controversial issue in the PCI community, partly because it has been affected by legacy busses so greatly, and partly due to the fact that latency and throughput have been somewhat at odds with each other, and high performance cards effectively want to optimize both. There are several components of latency, arbitration latency (which is typically 2 clocks for the highest priority device in the system); bus acquisition latency (the time from receiving a GNT# till the bus is IDLE and consequently available); and the target latency to respond to the transaction. Target latency for first data is now required to be 16 clocks; any more and the target must either perform a delayed transaction, or target-abort. (PCISIG, 2.1, 1994); Subsequent data transfers may see up to 8 clks to complete, but if the behavior of your target is known, the more accurate data would be factored into your calculations.

The controversial figure is that of bus acquisition latency which is very dependent on system configuration. The 2.0 specification recommends that 30µsec be the guideline for acquisition latency on the planar bus, while (PCISIG, 2.1, 1994) and (PCISIG, MM Guidelines, 1994) specify that bus levels off the planar bus typically would see less than 3µsec acquisition latency. This latency is affected by the value programmed into the master latency timer (MLT), as well as the typical and max burst duration of resources in the system. The number of masters and their priority also affect acquisition latency. For example, if there are 5 masters (max) on a bus level, each with their MLTs programmed to 32 clks, then if the target latency to first data was 16 clks and 8 clks between data transfers, then the acquisition latency assuming equal priority, could be as high as 32*(5-1)+8+16 = 4.56µsec of latency. The general rule is: tune your implementation to operate efficiently at 3µsec of latency, but insure that it can handle 30µsec of latency without catastrophic degradation. A few pixels dropped may be acceptable, while dropped frames or important control information are not.

Acquisition, generally the most variable and severe of the latencies, will require buffering in accordance with the following equation:

$$S_{lat\_buf} = t_{LAT} * f_{arrival} * W_{data}$$

where t_{LAT} is the max latency time, $f_{arrival}$ is the frequency of the data arriving to be transmitted (for master writes), and W_{data} is the number of bytes per farrival. Note that t_{LAT} should include the maximum latency that the current master's last transaction may introduce (8 PCLKs), as well as the initial target latency of the new master (16 PCLKS under the 2.1 guidelines). As an example, a 50MHz application clock delivering WORDs will require 30µsec*50Mhz*2B/WORD = 3K bytes (min) of total buffering, while a 10baseT interface would require 30µsec*10MHz*1/8 = 38 bytes of buffering (min). This does not include master latency (from IDLE to FRAME# asserted) nor target transfer latency.

After the bus is acquired, the sourcing FIFO must be able buffer data at least to the following depth:

$S_{xfer\_buf} = t_{tar\_xfer\_lat} *(f_{arrival} - f_{service}) * (W_{data}/4$ bytes/DWORD)

Obviously 3K bytes of on-chip buffering is expensive, and often impractical, especially when considering that substantially less buffering is required after the bus is acquired. An effective solution is to partition the buffering into levels: one transfer FIFO to provide data sufficient for the longest, fastest burst expected, and a buffer store to cover acquisition and initial target latencies. Synopsys' datapath buffers are tightly coupled with the master to not only initiate transfers from populated FIFOs, but also to manage exceptions. These buffers are able to cycle data at 0 wait-states during a burst, for popular, modern ASIC technologies. Their depth should be computed based on the master's maximum burst size and the relative difference in the master-slave bandwidth (S_{xfer_buf} equation).

Sitting behind the master datapath buffer can be a store that buffers the amount of data necessary to cover acquisition latency and initial target latency and provide storage (if necessary) for other resources. This buffer could be implemented in SRAM, or DRAM. Transfer from this buffer to the master datapath buffers could be performed by a simple DMA controller such as the one provided with the DesignWare™ MacroSet. Note that while transfers are in progress, the buffer store can be back-filling the master's datapath FIFO, providing for a resource sharing of storage, reducing area.

If the master's data arrival bandwidth is sufficiently low in portion to the acquisition latency and target bandwidth and latencies, then a single buffer may be sufficient (such as in the case of serial communications, etc.).

Buffer master writes has been the focus of discussion so far. Master reads need not account for acquisition latency, but only the classic queuing theory case of the difference in arrival rate versus service rate. If a master read requests as much as X bytes of data, and the transaction is completed without interruption, in the worst case, no data was serviced by the application, then X bytes of master read FIFO buffering would be required. A similar analysis can be applied to the slave read and write buffering requirements. The DesignWare™ MacroSet supports this by providing for parameterized depth FIFOs and/or registers each way, for master and/or slave. Furthermore, recent enhancements allow FIFOs to be constructed from Flip-Flops, or by leveraging ASIC vendor diffused or metal programmable SRAMs.

TEST DEVELOPMENT AND VERIFICATION

Ease of test development, modification and accuracy of tests and models are crucial in developing compliant systems in first-time silicon and in facilitating the development and validation of system tests. Models and tests provided in the Synopsys PCI kit used in this flow include the following:

Bus Functional Models (BFMs): These are BFMs of the PCI master and slave, as well as a passive monitor that observes and records bus activity and violations/exceptions and checks timing. Each model executes a series of commands test various compliance or user scenarios, and can therefore be used for complete system tests.

Compliance Suites: These are a series of verification suites derived in collaboration with the PCISIG Protocol subcommittee to verify functional accuracy. Scenarios based on functionality supported by the unit under test are automatically generated for the appropriate configurations. Automatic post-processing of the test results creates a compliance report showing which SIG tests passed, which failed and/or not applied.

Leveraging the test benches provided, one can create system-level tests by writing transaction sequences, as well as utilizing automatically generated test suites which validate protocol compliance. System tests and application tests can be stimulated and observed from the PCI bus through calls to these command procedures. Many sequences are generate which can be referenced. An example of such as test sequence is shown in Example 1.

CONCLUSION

In this paper, design and verification methodologies, design tradeoffs and models and tools have been presented which facilitate the rapid prototype and/or production design of efficient, compliant and reusable PCI-based systems. The authors welcome requests for additional design details.

```
- scenario_1.7 for multi-data phase retry cycle test
-- primary target behaves as PT (primary target)

-- Initialize target
    clear_delays;

-- memory cycles
                      -- fast
      config(0,12,0);          -- Transfer Limit = 0, abort limit = 12, Terminate
AFTER data
-- Target Reception of a Memory Write Cycle
      request(1,0,0);          -- request_limit = 1, decode = 0, delay = 0
      config(3,12,0);          -- Transfer Limit = 3, abort limit = 12, Terminate
AFTER data
--    Target Retry of a Memory Write Cycle
      request(1,0,0);

      config(0,12,0);          -- Transfer Limit = 0, abort limit = 12, Terminate
AFTER data
-- Target Reception of a Memory Read Cycle
      request(1,0,0);
```

Example 1 - Test Command Sequence Stub

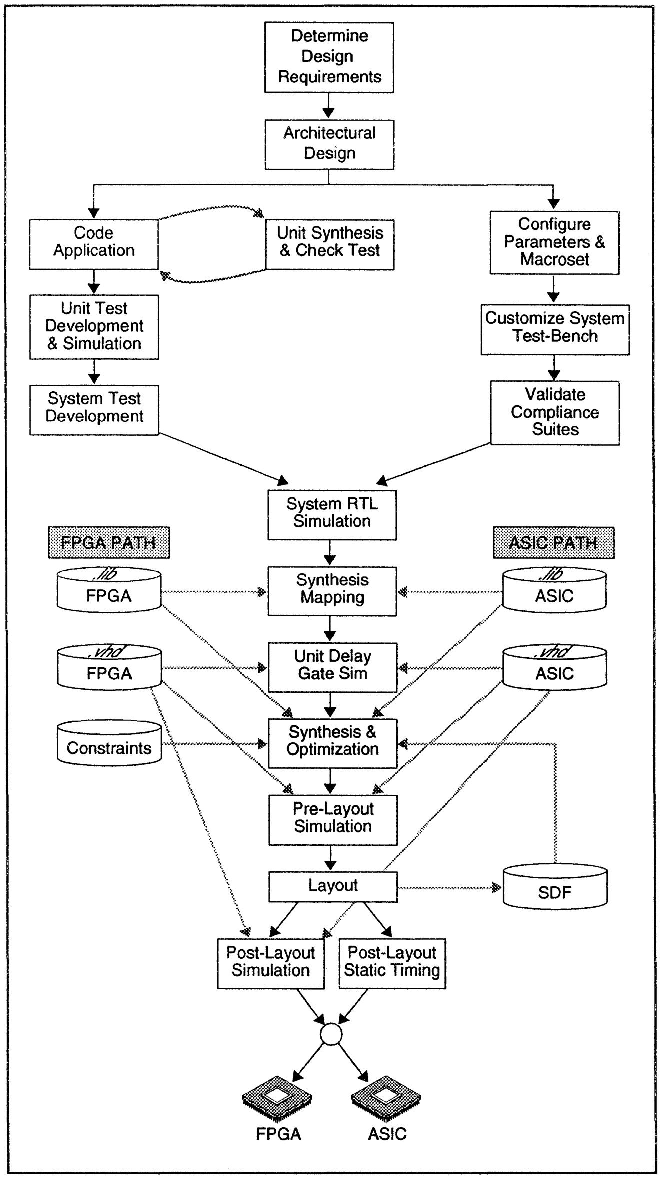


Figure 1 - High Level Design Flow For Implementation of PCI-based Systems

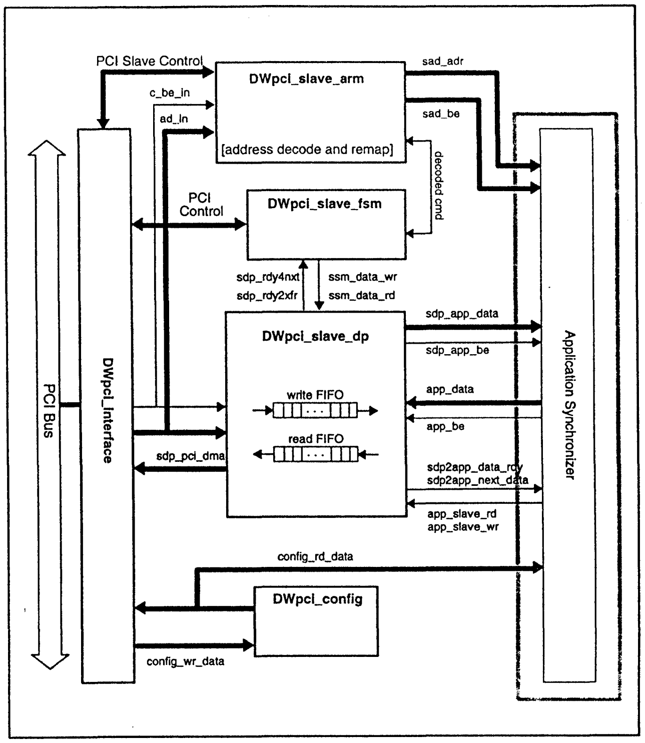


Figure 2 - Minimal Configuration

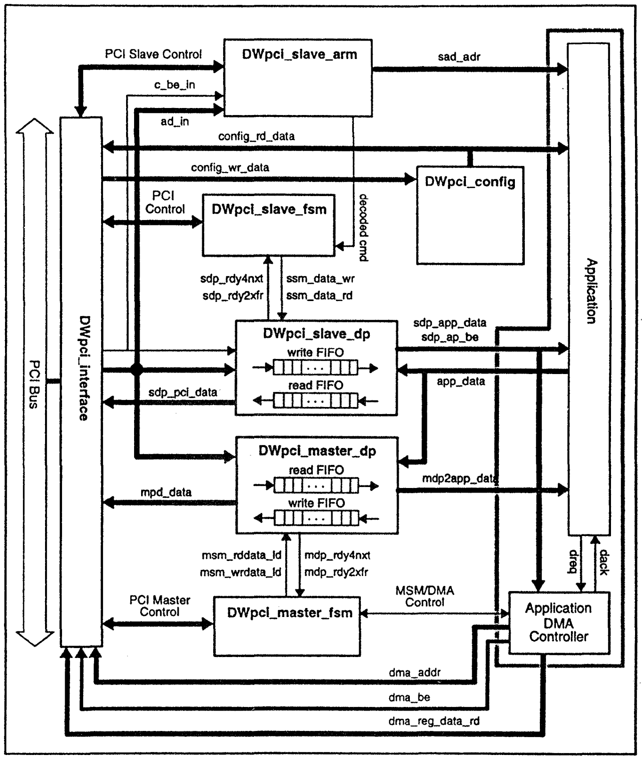


Figure 3 - Standard Master-Slave Configuration

REFERENCES

Synopsys®, Inc. DesignWare™ Components PCI MacroSet Databook. April 30, 1993.

Power Macintosh and PCI

David Limp
Power Macintosh Product Line Manager
Apple Computer, Inc.
One Infinite Loop
MS: 306-4PM
Cupertino, CA 95014
(408) 862-7218/974-2403 (fax)
limp@applelink.apple.com

In March of 1994, Apple Computer Inc. introduced the Power Macintosh line of computers, signifying a major leap forward in personal computer design and RISC technology. Customer acceptance of Power Macintosh has been phenomenal, with over one million units shipped in the first 10 months--well before Apple's one-year forecast. Support from third-party software and hardware vendors remains strong with over 500 native applications developed which have been optimized to take advantage of the PowerPC processor. New Power Mac applications continue to enter the market each week. Apple's transition from CISC-based technology to the RISC-based PowerPC chip has redefined the price/performance paradigm for personal computers and represents one of the most ambitious and successful accomplishments in Apple's history.

In 1995, Apple promises to extend the momentum and innovation of Power Macintosh by integrating two significant technologies into the next Power Macs: the PowerPC 604 processor and the Peripheral Component Interconnect (PCI) bus. The PowerPC 604 will bring even higher RISC performance to the desktop without requiring customers to pay workstation prices. The PCI bus in Power Mac systems will allow Apple customers to use industry-standard PCI cards, thus enabling new solutions and higher performance on the Power Macintosh platform.

Apple is committed to the PCI bus for the long term. The company has adopted Open Firmware as its boot interface to allow for a very open implementation of PCI cards on the Power Macintosh. Any PCI card that complies with the PCI 2.0 specification should work on Power Mac systems with PCI, thus making it very easy for developers to offer products for the Macintosh platform. The session will present an overview of why Apple chose PCI as its future bus strategy and how developers can take advantage of new PC and Macintosh markets where traditional PCs did not have significant market share.

PCI & Open Firmware™

Lillian Leung
Senior Software Engineer
FirePower Systems
190 Independence Drive
Menlo Park, CA 94025
Ph. (415) 462-6217 Fax (415) 462-3051

The PCI bus has been adopted on the Intel PC/AT platforms and non-Intel platforms. The PowerPC PReP platform is what I am involved with currently. Prior to that I'd been involved with a Pentium-based PCI/EISA server. With my background as a firmware engineer, I would like to concentrate on the future of PCI in terms of firmware and software support. I would like to encourage PCI device vendors to develop Fcode drivers for Open Firmware (IEEE 1275 Standard). After all, Open Firmware runs on PowerPC PReP, PowerPC Apple, Intel, Sparc and other machines. By providing Fcode drivers in device ROM, the PCI device vendors will expand their market shares beyond the legacy PC\AT. New markets can be penetrated with ease. It's a win-win situation.

PCI's Role in Mobile Computing

Randy Giusto
BIS Strategic Decisions
One Longwater Circle
Norwell, MA 02061
Phone (617) 982-9500
Fax (617) 982-1724

PCI has the potential to deliver true desktop power to mobile computing. Today, the mobile professional has the processor power, memory, and storage features that desktops provide, but lacks a bus to provide true performance for applications such as multimedia. PCI, along with cardbus, will provide users more power for emerging applications and capabilities as we move from a text-based metaphor to a visual one.

LATE SUBMISSIONS

Windows 95 and PCI

Marshall Brumer
System Developer Relations Group
Microsoft Corporation
One Microsoft Way
Building 6
Redmond, WA 98052-6399
(206) 936-5840/7933 (fax)
marshalb@microsoft.com

This talk covers the areas within PCI in which Microsoft has encountered issues and has created solutions for within Windows 95 and Windows NT. Issues and problems are explained and actionable solutions given.

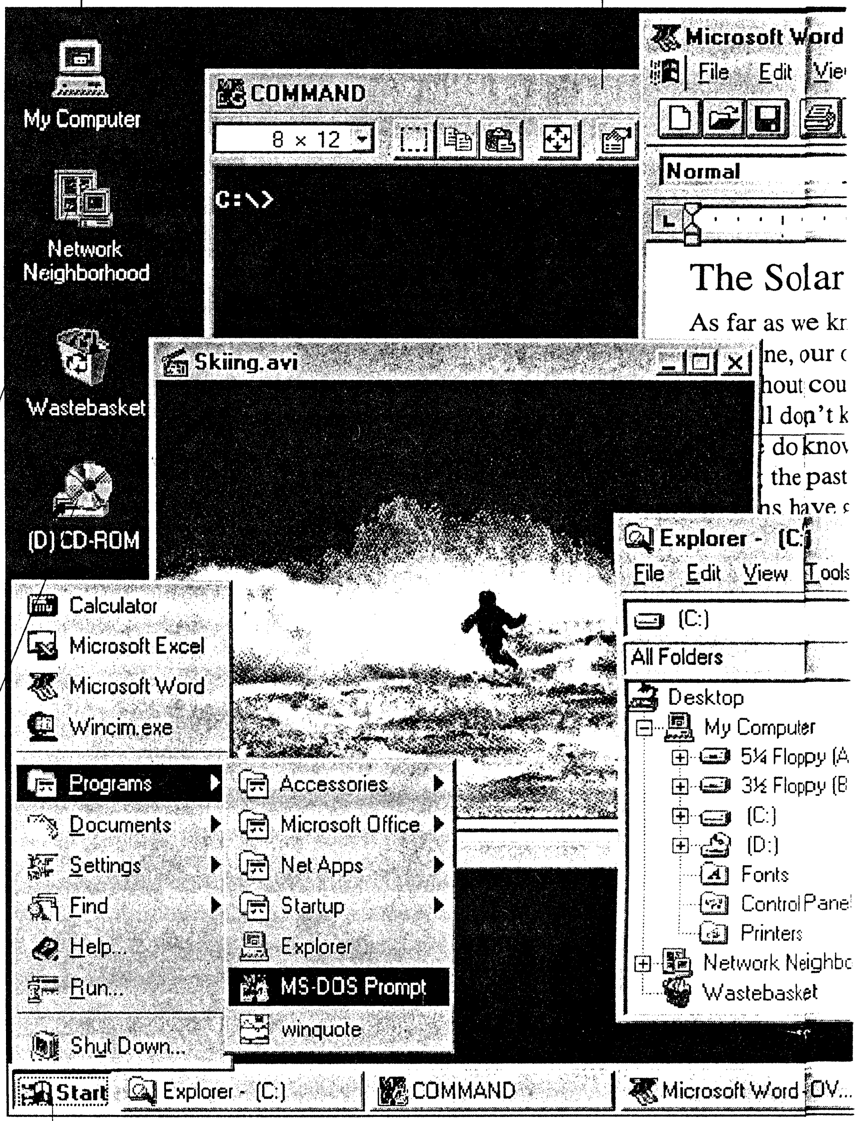
Paper reprinted courtesy of Microsoft Magazine, Winter 1994.

Special preview: Windows 95

You've heard about the next major release of Windows. Now, here's a look at what's in it for you.

COMING 1st HALF OF 95

THERE'S MORE POWER in your personal computer than you may think. The upcoming version of the Microsoft Windows operating system—Windows 95—replaces MS-DOS®, Windows version 3.1, and Windows™ for Workgroups 3.11 with a new system that takes advantage of your computer's power to let you do more tasks faster and with greater ease. The minimum hardware configuration for Windows 95 is a 386DX computer with just 4 megabytes (MB) of memory. With Windows 95, you can run your existing 16-bit programs as well as new 32-bit applications.



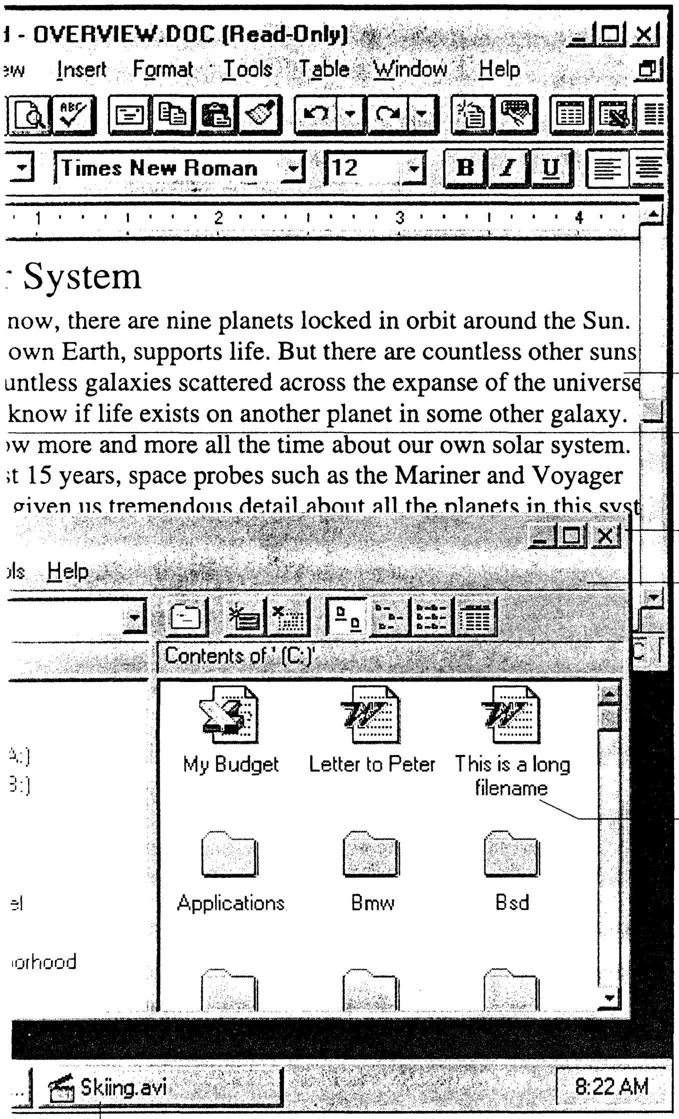
So long, Program Manager
Now, icons for your programs, folders, and files can sit directly on your desktop. So the last thing you worked on is right there, ready to go. Or, click My Computer to see folders of all your files.

Just click to connect
Click Network Neighborhood to access what's on the network. Windows 95 is also an easy introduction to the Internet. Built-in software uses your modem to dial up and connect.

No more growing pains
Plug and Play means you don't have to select drivers and settings manually or run a configuration program to add, set up, or change computer hardware. Just plug in a CD-ROM drive, more memory, or a network card, and Windows 95 detects and configures it for immediate use—automatically.

Start here
Click Start for immediate access to what you want to do. Even if you care never to learn anything else about Windows 95, you can get to 99 percent of the tools you'll ever use just by clicking Start.

Long live MS-DOS
Windows 95 replaces the Microsoft MS-DOS operating system but has complete backward compatibility with it, so you can run any MS-DOS–based application you like.



: System

now, there are nine planets locked in orbit around the Sun.
own Earth, supports life. But there are countless other suns
untless galaxies scattered across the expanse of the universe
know if life exists on another planet in some other galaxy.
ow more and more all the time about our own solar system.
st 15 years, space probes such as the Mariner and Voyager
given us tremendous detail about all the planets in this syst

Do two, three, four things at once
View a video while printing a document in Word while copying a disk. Multitasking makes it possible—and easy.

Easy to explore
Explorer shows much more than File Manager. Everything on your computer, from documents to programs, is laid out both hierarchically and pictorially in the Windows 95 two-pane Explorer. Just click the pictorial image, and the toolbar displays buttons that work with the object represented by the picture.

Get wordy
Filenames can be more descriptive because you can use up to 255 characters. If you want, call a file Everything I ever wanted to do with my office except that I haven't found the time.

What's up?
See exactly what's running on your computer. The task bar shows what you can switch to—with just a click.

The Microsoft Network: Microsoft's own online service

When you upgrade to Microsoft Windows 95, you can merge onto the information highway while you're at it. Windows 95 includes access to a new, easy-to-use online service, The Microsoft Network, that puts you in touch with the entire world. And it's your direct connection to Microsoft.

With The Microsoft Network, it will be easy to get information or support on Microsoft products straight from the source. In addition, access to many top computer hardware and software companies will be available through the service. The Microsoft Network also includes an array of bulletin boards where you can exchange ideas with those who share your interests—whether that means debating the merits of the latest Oliver Stone film or getting advice on how to grow prize tomatoes. Plus, you can tap into thousands of Internet discussion groups. And, The Microsoft Network provides you with electronic mail so you can send messages to other members of The Microsoft Network and to anyone who has an Internet address.

If you're interested in going online with Microsoft, watch for the free trial offer during your setup of Windows 95.

You can keep up to date on Windows 95 with our free electronic newsletter. To subscribe, send Internet mail to enews@microsoft.nwnet.com with the words SUBSCRIBE WINNEWS in the body of your message. If you are a CompuServe customer, send mail to INTERNET:enews@microsoft.nwnet.com.

When you subscribe to the *WinNews* newsletter, you will receive a welcome message. Because of the popularity of the newsletter, however, the WinNews computer has experienced some capacity shortages. If you don't receive this welcome message within three days, please resubmit your subscription request. M

IBM27-82351 PCI TO PCI BRIDGE

Robert Kilmartin, Alvar Dean, Marc Faucher / Uri Elazar, Ophir Nadir
IBM Microelectronics / IBM Israel
1000 River Street, Essex Jct., VT 05452

ABSTRACT

In today's computer environment, the PCI local bus standard has emerged as the leading high performance bus of choice. It provides a high bandwidth, processor independence, low latency transfer protocol with a built-in configuration mechanism to facilitate inter-operability. The high speed nature and electrical requirements of the PCI bus result in a practical limit of around ten PCI loads. A PCI to PCI bridge significantly increases PCI bus expandability. This enables motherboards to increase the number of on-board PCI devices and add-in card connectors, and allows adapter cards to offer multiple PCI device solutions.

This paper presents IBM's first solution in the PCI to PCI bridge market. The IBM27-82351 is a fully compliant PCI to PCI bridge with many features that enhance the performance and usability beyond the basic requirements of the PCI to PCI Bridge Specification 1.0. The IBM27-82351 bridge provides several performance enhancing features such as reduced latency, data combining, improved buffer management, and arbitration control. In addition, enhanced address decode and forwarding control is provided via configurable address registers. These features and others will be fully presented.

PCI BUS ANALYZER GREATLY SIMPLIFIES TEST & DEBUGGING OF PCI SYSTEMS

Thomas Nygaard
Vice President, VMETRO
Tel. (713) 584-0728, Fax: (713) 584-9034
Email: thomasny@vmetro.oslonett.no

Abstract

Testing and debugging PCI bus systems can be a challenge, not only because of the tough specifications that require careful timing design of the hardware, but just as much on the software side, when several complex devices must play together with potential configuration and initialization problems, hardware and software incompatibilities, problems byte-swapping, interrupts etc.

The common factor for most of these problems is that they relate to interactions between chips or boards that all reside on the PCI bus. This means that observing the activity on the PCI bus is the key to finding and solving problems. For this reason, VMETRO is offering a PCI Bus Analyzer that greatly simplifies test & debugging of PCI systems.

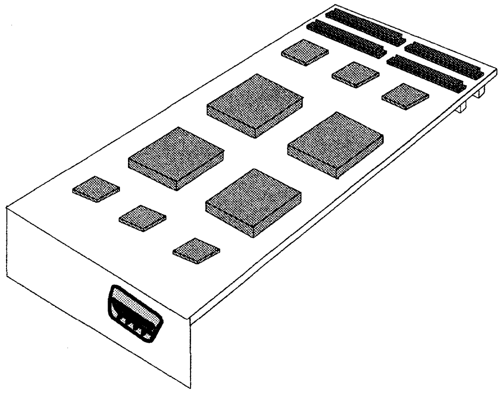


Figure 1. A full-featured Logic Analyzer for PCI on a PMC - PCI Mezzanine Card

THE PBT-315 PCI BUS ANALYZER

The PBT-315 is an advanced self-contained Bus Analyzer for the PCI bus, implemented as a single-width PCI Mezzanine card (PMC). This allows the analyzer to be plugged directly into a PCI motherboard like a PCI bus CPU card with PMC slots. Alternatively, the analyzer can be delivered with a buffered PCI/ISA shaped adapter board that plugs into desktop systems like PCs or workstations.

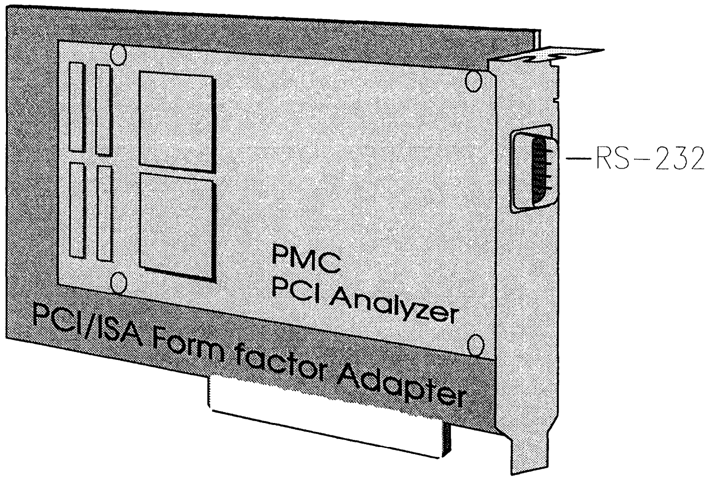


Figure 2. Also for analysis of desktop PCI systems using the PCI/ISA form factor adapter.

POWERFUL DEBUGGING FEATURES

The PBT-315C State, Timing and Statistics Analyzer includes a number of powerful features which have been chosen to make the analyzer as useful as possible in detecting complex problems. The most important features are :

- **State, Timing & Statistics Analysis on all channels.**
- **Up to 50MHz sampling rate using CLK- or transfer-synchronous sampling.**
- **128 sampling channels for 32 & 64-bits PCI support.**
- **64K Trace Memory depth.**
- **4 full-speed Word Recognizers with Range, Don't care and NOT.**
- **16 levels Trigger/Qualifier Sequencer.**
- **Trigger after Delay or Event Count.**
- **Demux'ed Address, Command and Data.**
- **Time Tags in Trace Buffer show time between samples and Wait states.**

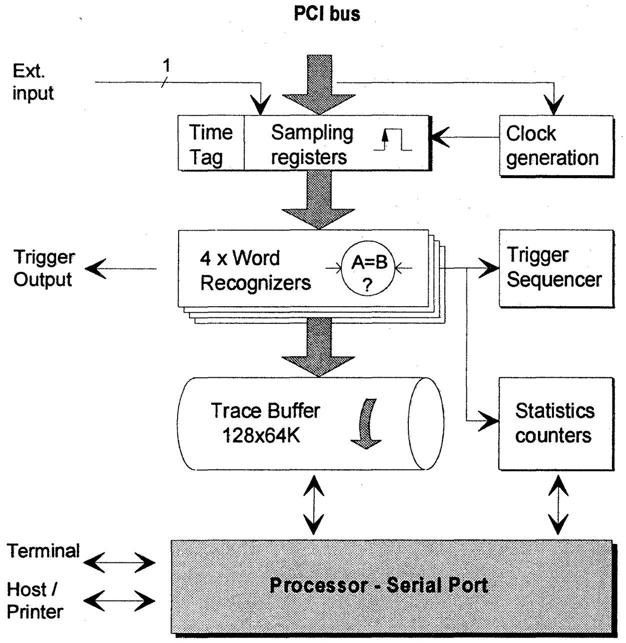


Figure 3 : Block Diagram of PBT-315 Bus Analyzer

TRACE DATA PRESENTATION WITH DEMULTIPLEXED ADDRESS / DATA

The powerful features of the PBT-315 Bus Analyzer mentioned above allows the capture of a comprehensive set of information representing the activity on the bus. One very important feature is the capability to demultiplex address, commands and data into separate trace channels. This not only simplifies readability of the trace, but allows powerful triggers involving both address and data to be defined easily.

The captured data are presented to the user in a uniform and easily understandable way. Trace data can be captured and presented in the form of an alphanumeric trace list, where data is sampled either on each CLK edge or only when there is a valid data item transferred. Alternatively, data can be shown as a waveform diagram.

Regardless of the type of presentation selected, the user can scroll forward and backward in the trace data, and can also select which signals shall be present on the screen. This allows the user to extract the maximum of relevant information from the trace data with a minimum of effort.

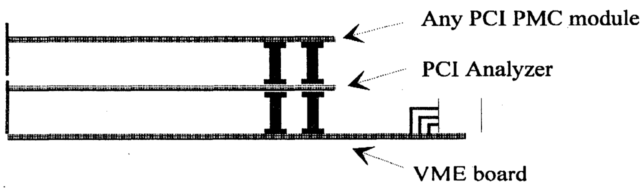


Figure 4. In PMC systems, the analyzers allow stacking of a PMC module on top of the analyzer.

STACKABLE PMC

Being equipped with both male and female PMC connnectors, one standard PMC module card may be piggybacked on the PBT-315, eliminating the need for a spare PCI slot. The analyzer may operate from a separate power supply, and the analyzer and the piggybacked PMC module under test have isolated, separate reset functions to eliminate the need for time-consuming host reboot during debugging or production testing of PMC modules.

STATE ANALYSIS WITH COMPLEX TRIGGERING

State analysis is the process of capturing events or cycles one by one at the speed they occur in the system. This technique is typically used to debug problems associated with software. To trigger on even the most complex of situations the PBT-315 analyzer is equipped with 16 triggering levels, and with 20-bit event counters, allowing up to 1M occurrences of an event in the trigger program. Delay counters are also included, providing programmable delays anywhere in the triggering sequence. This is particularly useful in real-time systems. Figure 4 shows an example of a complex trigger condition.

```
If (Event 0) then
   Count 1048575 occurrences of (DMA-READ)
   Store (Event 0 or Event 1)
   If (Event 0) then
      Count 65535 occurrences of (Event 1)
      If (BUSERROR) then
         Delay 120us then
         Trigger at 75% of Trace
      Else
         Restart
           .
           .
           .
   (Up to 16 levels)
```

Figure 5 : Example of how to utilize multiple count and delay statements to form a complex trigger condition.

Sometimes a problem calls for triggering or storing on cycles that comply with a number of conditions except one, like in the two examples shown in figures 5 and 6. The first of these examples can be used to trigger when a subroutine returns an unexpected value, but one has no idea what that value may be. In the second example one might be able to reveal who is responsible for data being overwritten in a byte-oriented device buffer by using the analyzer to capture only data of "not byte" size (i.e. word, lword), typically written by a program gone wild. These examples show that the NOT capability gives a new level in flexibility in specifying the desired trigger point or filter specification.

```
If (Write to Address X, Data NOT equals 0) then ...

Store (Write to Address Range X-Y, Data of Size NOT
Byte)
```

Figure 6 : The possibility to add NOT statements in signal groups in the word recognizers increases triggering and filtering flexibility considerably.

INVESTIGATION OF A SOFTWARE PROBLEM

The PBT-315 analyzer can be of great assistance in investigating certain types of software errors in a PCI bus system, especially those kind of errors where one or more boards fail to implement some kind of software protocol correctly. In these cases a clear view of the traffic on the PCI bus may identify not only what kind of error occurred, but also which board caused it and how.

One example would be to investigate a problem where three PCI bus masters are synchronising their actions by communicating through a common mailbox. The contents of this mailbox may occasionally be destroyed, presumably because one of the masters under certain conditions uses the wrong data type to access it.

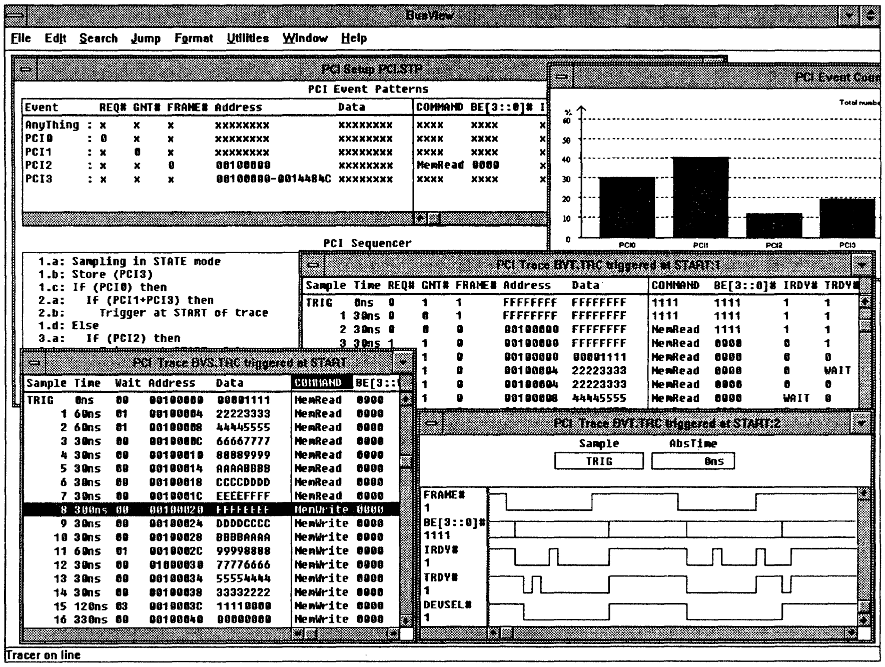
One may in such a case set up a trigger condition that captures accesses within a specific area (representing the mailbox) and with wrong data type. The trace output will easily show that we can identify not only the access itself and the address accessed in the mailbox, but also which PCI bus master performed the access. An inspection of the trace data preceding the trigger may even tell us exactly why, or at least under exactly what conditions, this error occurred.

INVESTIGATION OF SYSTEM PERFORMANCE USING STATISTICS FUNCTIONS

Finally we may use the PCI anallyzer to look at the performance of a PCI bus system. In order to simplify such an investigation the PBT-315 Bus Analyzer system is equipped with a Statistics module. This module allows the user to gather many different kinds of data as to how the traffic on the PCI bus behaves and to spot uneven distribution of system load and other symptoms that may represent performance bottlenecks.

WINDOWS OR TERMINAL-BASED USER-INTERFACE

The PBT-315 can be operated from a terminal, PC/WS with terminal emulator or from a true Windows application; VMETRO's BusView for Windows. In either case the user will find similar windows, command bars, pull-down menus and dialog boxes.


BusView
File Edit Search Jump Format Utilities Window Help
PCI Setup PCI.STP
PCI Event Count
Total numbe
%
60
50
40
30
20
10
0
PCI0
PCI1
PCI2
PCI3
PCI Trace EVT.TRC triggered at START:1
PCI Trace BVS.TRC triggered at START
PCI Trace EVT.TRC triggered at START:2
Sample
AbsTime
TRIG
0ns
FRAME#
1
BE[3::0]#
1111
IRDY#
1
TRDY#
1
DEVSEL#
1
Tracer on line

```
PCI Event Patterns
Event       REQ# GNT# FRAME# Address           Data      COMMAND  BE[3::0]# I
AnyThing  : x    x    x      xxxxxxxx          xxxxxxxx  xxxx     xxxx      x
PCI0      : 0    x    x      xxxxxxxx          xxxxxxxx  xxxx     xxxx      x
PCI1      : x    0    x      xxxxxxxx          xxxxxxxx  xxxx     xxxx      x
PCI2      : x    x    0      00100000          xxxxxxxx  MemRead  0000      x
PCI3      : x    x    x      00100000-0014484C xxxxxxxx  xxxx     xxxx      x

PCI Sequencer
  1.a: Sampling in STATE mode
  1.b: Store (PCI3)
  1.c: If (PCI0) then
  2.a:   If (PCI1+PCI3) then
  2.b:     Trigger at START of trace
  1.d: Else
  3.a:   If (PCI2) then
```

```
Sample Time REQ# GNT# FRAME# Address   Data      COMMAND  BE[3::0]# IRDY# TRDY#
TRIG   0ns  0    1    1      FFFFFFFF  FFFFFFFF  1111     1111      1     1
   1 30ns   0    0    1      FFFFFFFF  FFFFFFFF  1111     1111      1     1
   2 30ns   0    0    0      00100000  FFFFFFFF  MemRead  1111      1     1
   3 30ns   1    1    0      00100000  FFFFFFFF  MemRead  0000      0     1
                 1    0      00100000  00001111  MemRead  0000      0     0
                 1    0      00100004  22223333  MemRead  0000      0     WAIT
                 1    0      00100004  22223333  MemRead  0000      0     0
                 1    0      00100008  44445555  MemRead  0000      WAIT  0
```

```
Sample Time  Wait Address   Data      COMMAND   BE[3::
TRIG    0ns  00   00100000  00001111  MemRead   0000
   1  60ns   01   00100004  22223333  MemRead   0000
   2  60ns   01   00100008  44445555  MemRead   0000
   3  30ns   00   0010000C  66667777  MemRead   0000
   4  30ns   00   00100010  88889999  MemRead   0000
   5  30ns   00   00100014  AAAABBBB  MemRead   0000
   6  30ns   00   00100018  CCCCDDDD  MemRead   0000
   7  30ns   00   0010001C  EEEEFFFF  MemRead   0000
   8 300ns   00   00100020  FFFFEEEE  MemWrite  0000
   9  30ns   00   00100024  DDDDCCCC  MemWrite  0000
  10  30ns   00   00100028  BBBBAAAA  MemWrite  0000
  11  60ns   01   0010002C  99998888  MemWrite  0000
  12  30ns   00   01000030  77776666  MemWrite  0000
  13  30ns   00   00100034  55554444  MemWrite  0000
  14  30ns   00   00100038  33332222  MemWrite  0000
  15 120ns   03   0010003C  11110000  MemWrite  0000
  16 330ns   00   00100040  00000000  MemWrite  0000
```

CONCLUSION

In this paper we have discussed how the PBT-315 Bus Analyzer system can be a very powerful tool in detecting, locating and fixing different kinds of PCI bus system problems.

Altogether, the PBT-315 Bus Analyzer constitutes a very important tool when building and integrating PCI bus based systems. In many cases this tool could save a very considerable amount of time in debugging such systems, as well as greatly simplify the issue of quality testing.

PCI, A TECHNOLOGY ENABLER

Sean Burke
Director, Desktop Product Marketing and Planning
Dell Computer Corp.
2214 West Breaker Lane
Austin, TX 78758
512-728-4984

PCI technology is a key technology that impacts various customer segments today and will provide industry-leading solutions for tomorrow. As leader of dell's development plans, Mr. Burke will discuss how Dell has integrated PCI technology across its desktop product lines and outline how Dell has implemented PCI in various ways, according to its diverse target markets -- ranging from small to medium sized businesses, individul enthusiasts to large corporations.

For enthusiasts and small-to-medium sized businesses, PCI provides a flexible platform enabling users to achieve maximizing performance from their system.

Large corporate customers depend on stability, standardization, and reliability. Mr. Burke will highlight how Dell has integrated PCI components on the system board, such as int4egrated Pci SCSI and integrated PCI video. With these implementations, Dell delivers both stability and cost efficient computing to corporate customers.

As a leader in the PCI industry, Dell strives to be on the leading edge of important technology transitions -- delivering relevant technology as it becomes available. PCI is an enabler for other key developing standards, such as video architectures and chip design. Fast to market is the cornerstone of Dell's direct model and PCI helps the company stay on the leading edge of performance.

A Programmable Logic Design Approach to Implementing PCI Interfaces

Martin Won
Senior Applications Engineer

Glen Quiro
Applications Engineer

Altera Corporation
3 W. Plumeria
San Jose, CA 95134
800-9-ALTERA

This paper will discuss a programmable logic approach to implementing Peripheral Component Interconnect (PCI) bus interfaces. PCI is rapidly gaining popularity for its high performance and wide bandwidth, and in order to take full advantage of its capabilities, system designers must consider a number of possible implementations. The portion of the PCI bus scheme that this paper will address is the the interface between the PCI bus itself and any back-end function that needs to use the bus, either to send or receive data.

A programmable logic implementation of a PCI interface offers several options that non-programmble logic implementations (i.e. chip sets) do not. The most attractive aspect of using programmable logic for PCI bus interfacing is the flexibility of the implementation. Programmable logic provides the flexibility to customize the interface to the back-end function. There is also the capability to easily change or alter the interface design to update or add features to the overall product. Also, programmable logic features the option to incorporate portions of the back-end function into the programmable logic device itself (if resources are available; see the *Hardware Implementation* section of this paper), thus conserving board real estate. Finally, another reason for choosing programmable logic over a less-flexible solution is that a dedicated solution might not support all the possible bus cycles specified in the PCI specification, whereas programmable logic is open to support all the existing bus cycles, plus any that may be defined in the future.

Customizable Functionality

There are a number of areas in a PCI interface that need to be tailored to suit the needs of the function that is being interfaced to a PCI bus. This tailoring ranges in complexity from choosing not to implement certain functions (i.e. parity check and/or parity error) to fine-tuning the logic to meet critical needs (i.e. limiting the response of the control state machine to certain bus cycles to optimize the timing). These and other types of changes are easily made in a programmable logic design approach with straightforward modifications of the design description. Specific modifications will be discussed in the section of this paper titled *Modifying/Customizing the Macrofunctions*.

Description of PCI Macrofunctions

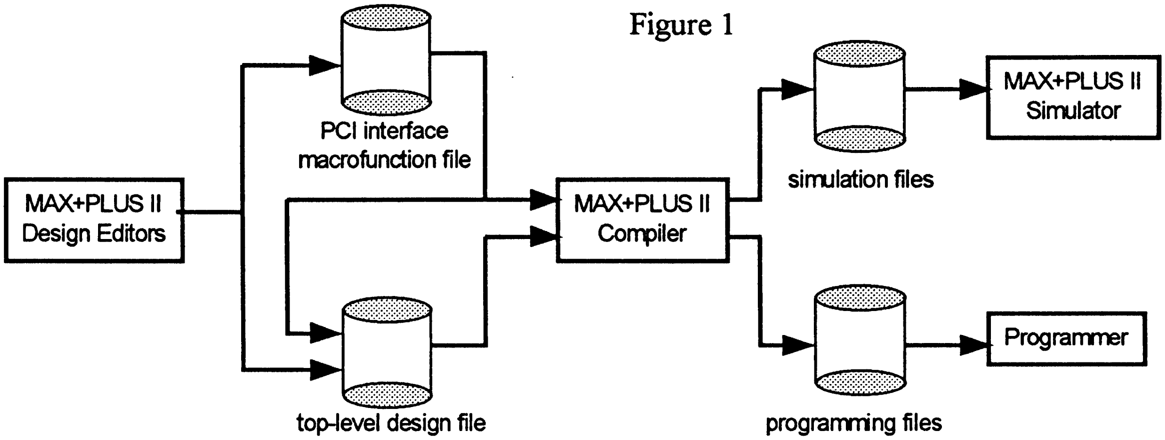
A set of PCI interface designs has been created for use with Altera's programmable logic devices. These designs (or *macrofunctions* in Altera's terminology) are meant to serve as the foundations for a PCI interface design, with the designer changing aspects of the macrofunction and adding/removing components to suit the individual needs of the product. At present, there are three macrofunctions: a master interface, a target interface, and a parity generator. A macrofunction for a combined master/target interface is in development.

Several Altera devices are specified by the PCI Special Interest Group (SIG) as being PCI-compliant, including many members of the MAX 7000 and FLEX 8000 families. A complete list of these devices is available both from the PCI SIG and from the Altera Marketing department at (408) 894-7000. There is also a complete checklist of items that are associated with PCI compliance; for more information on specificities of Altera's device compliance, consult Altera's *Application Brief 140: PCI Compliance of Altera Devices*.

The PCI macrofunctions have been described using Altera Hardware Description Language (AHDL). In this form, they are ready to be incorporated into any design targeted for an Altera programmable logic device. The development tool used to design for Altera devices is MAX+PLUS II, a complete development environment including design entry, compilation, and simulation capabilities, as well as interfaces to most popular CAE tools. The rest of this section describes MAX+PLUS II operation; readers who are familiar with MAX+PLUS II but not AHDL may wish to skip ahead to the *Brief Introduction to AHDL.* Readers who are familiar with MAX+PLUS II and AHDL will probably want to skip forward to *Modifying/Customizing the Macrofunctions*.

Within the MAX+PLUS II design environment, macrofunctions can be used either as stand-alone design descriptions or as part of larger design descriptions. Depending on the design requirements, the designer can modify the design description of the appropriate PCI interface macrofunction, or instantiate the macrofunction into a larger design description. Design descriptions can be composed of any combination of graphics, text, and waveform design files. A completed design description is submitted to the MAX+PLUS II Compiler, which produces programming and simulation files for the targeted programmable logic device. Simulating the design using timing information from the overall system contributes to guaranteeing the reliable operation of the device in the system. The MAX+PLUS II design flow (modified to show use of a PCI macrofunction) is illustrated in Figure 1 below:

Figure 1



Brief Introduction to AHDL

While this document cannot include a full treatment of AHDL, an understanding of some of the basic concepts of AHDL will enable a designer to make most of the changes necessary to use and customize the PCI macrofunctions. To this end, a few simple AHDL examples will be discussed in this section. Readers who are familiar with AHDL can skip forward to the *Understanding and Customizing the Macrofunctions* section. For a complete treatment of AHDL consult the MAX+PLUS II AHDL manual as well as MAX+PLUS II On-Line Help.

AHDL is a text-based design language in which the behavior of the desired logical function is described. For example, Figure 2 shows an AHDL fragment of the parity generator (a complete AHDL description of this macrofunction is available as part of Altera's PCI Design Kit or directly from Altera's Applications group). Note that the bit widths of the input buses (address or `ad` and command/byte enable or `c_be`) are indicated by the range delimiter `[X..Y]` where X and Y determine the upper and lower bound of the bus width. Note also that the parity signals `par0` and `par1` are generated via boolean equations, where the symbol `$` corresponds to the logical XOR operation.

```
SUBDESIGN pci_par
(
    ad[31..0], c_be[3..0]      :    INPUT;
    parity                     :    OUTPUT;
)

VARIABLE
    par[10..0]                 :    NODE;

BEGIN

    -- Parity generation equations

    par0 = ad0 $ ad1 $ ad2 $ ad3;
    par1 = ad4 $ ad5 $ ad6 $ ad7;
```

```
•
•
•
```

Figure 2

The AHDL fragment in Figure 3 below illustrates the declaration of the Base Addess Registers (BAR):

```
SUBDESIGN target
(

-- PCI Interface Signals

     CLK            : INPUT;   -- PCI Clock
     AD[31..0]      : BIDIR;   -- Multiplexed address/data
     RST            : INPUT;   -- PCI Master Reset
•
•
•
)

VARIABLE

     BAR[31..5]     : DFF;     -- Base Address Registers

BEGIN

     BAR[].clk  = CLK;
     BAR[].clrn = RST;
     BAR[31..5].d = Write_BAR & AD[31..5] # !Write_BAR &
     BAR[31..5].q;
•
•
•
```

Figure 3

The AHDL fragment in figure 3 also illustrates the description of the logic required to write a value into the Base Address Registers. The last line of AHDL in the fragment (shown below) defines that the 28-bit value to be placed on the d inputs of the BAR is the value on the address lines (`AD[31..5]`) logically ANDed with the binary signal `Write_BAR` (defined outside of this fragment) OR the value from the q outputs of the BAR anded with the complement of `Write_BAR`.

```
     BAR[31..5].d = Write_BAR & AD[31..5] # !Write_BAR &
     BAR[31..5].q;
```

The last item of interest in both AHDL fragments is the use of the two sequential dashes to indicate a comment. This notation can also be used to prevent lines of text in an AHDL design description from being compiled into the hardware implementation of the design. For example, if a designer did not wish to include a PCI master rest input signal (listed as RST in figure 4) in the design, he or she could add two dashes to the beginning of that line as indicated in figure 4 below:

```
SUBDESIGN target
(

-- PCI Interface Signals

     CLK              : INPUT;  -- PCI Clock
     AD[31..0]        : BIDIR;  -- Multiplexed address/data
--   RST              : INPUT;  -- PCI Master Reset
•
•
•
)
```

Figure 4

The other means of commenting out a line of AHDL code is with the percent symbol (%). Unlike the sequential dashes, use of a single percent sign indicates the beginning of a comment, while the second percent sign indicates the end of the comment. For example, if a designer wished to comment out the last two lines of the AHDL fragment in Figure 4 using percent signs, the resulting text would look like Figure 5:

```
SUBDESIGN target
(

-- PCI Interface Signals

     CLK              : INPUT;  -- PCI Clock
%    AD[31..0]        : BIDIR;  -- Multiplexed address/data
     RST              : INPUT;  -- PCI Master Reset      %
•
•
•
)
```

Figure 5

AHDL designs are saved as files with a .tdf (Text Design File) extension. MAX+PLUS II recognizes files with the .tdf extension as AHDL design files to be compiled or incorporated into designs for Altera programmable logic devices.

Modifying/Customizing the Macrofunctions

There are a number of ways a designer might customize the PCI macrofunctions to suit the needs of a particular design. This section of the paper will describe a few of them. The whole range of possible variations on a PCI interface design can not, of course, be encapsulated into any single document, but the intention of covering a few examples here is to convey the effort involved in such changes. The customizations that will be discussed are:

(1) Adjusting the width of the address and data buses connecting the interface to the back-end function
(2) Including/excluding a parity check/parity error function

Other customizations that will be discussed (in somewhat less detail) are:

(1) Including some or all of the Configuration Space in the PLD
(2) Generating signals for the back-end function

The means for customization will be modification of the AHDL design files using any standard ASCII text editor. In this paper, the design file referenced will be the design for the target interface for Altera's product-term based devices. This file is called TAR_MAX.TDF.

Adjusting the Width of Address/Data Buses

The width requirement for the address and data buses connecting the PCI interface and the back-end function vary with the needs of the back-end function. Changing these widths requires three modifications to the AHDL design.

(1) The number and names of the device pins corresponding to these buses must be changed to fit the desired number and names of the signals.

(2) The number of registers that hold the address and/or data information must be modified accordingly.

(3) The number of tri-state buffers that control the passage of the address and/or data information to the outside world must be changed to correspond to the new number of address and/or data lines.

In the TAR_MAX.TDF file, signals that connect to the outside world (via device pins) are defined in the first part of the Subdesign section. This section is excerpted in Figure 6 below:

```
SUBDESIGN tar_max
(

-- PCI Interface Signals
```

```
  CLK            : INPUT;  -- PCI Clock
  AD[31..0]      : BIDIR;  -- Multiplexed address/data
  C_BE[3..0]     : INPUT;  -- Command/Byte enable
  PAR            : BIDIR;  -- Parity
  PERR           : BIDIR;  -- Parity Error
  SERR           : OUTPUT; -- System Error
  FRAME          : INPUT;  -- Transfer Frame
  IRDY           : INPUT;  -- Initiator Ready
  TRDY           : BIDIR;  -- Target Ready
  DEVSEL         : BIDIR;  -- Device Select
  IDSEL          : INPUT;  -- ID Select
  RST            : INPUT;  -- PCI Master Reset
  STOP           : BIDIR;  -- Stop Request

-- Interface Back-End Device Signals

  Addr[7..0]     : OUTPUT; -- Address From Device
  Data[31..0]    : BIDIR;  -- Data To/From Device
  -- Dpar        : INPUT;  -- Data Parity From Device
  BE[3..0]       : OUTPUT; -- Configuration Byte Enables
  Dev_req        : INPUT;  -- Request From Device
  Dev_ack        : OUTPUT; -- Transfer Complete Ack.
  Rd_Wr          : OUTPUT; -- Read/Write
  Cnfg           : OUTPUT; -- Configuration Cycle
  T_abort        : INPUT;  -- Fatal Error has occured
  Retry          : INPUT;  -- Target signaled a retry
  Reset          : OUTPUT; -- PCI Reset
)
```

Figure 6

The address and data buses to the back-end function are the two lines (bolded) directly underneath the comment line "`Interface Back-End Device Signals`". Note that the keyword `OUTPUT` after the colon indicates that the "objects" declared by the name `Addr[7..0]` are output pins.

The first step to modifying the width of these buses is to change the number ranges in the brackets following the names of the signals. For example, the address bus (shown as being 8 bits in width) can be modified to be a 4-bit bus by changing the line:

```
    Addr[7..0]     : OUTPUT; -- Address From Device
```

to

```
    Addr[3..0]     : OUTPUT; -- Address From Device
```

Likewise, the data bus `Data[31..0]`can be modified to any width with a similar operation. Note that the data bus signals are defined to be of type `BIDIR`, indicating that they are bidirectional signals.

The second step is to change the number of registers that hold the address and/or data information and before going to the tri -state buffers. The registers for the address information are called Addr_reg. The line of AHDL in the TAR_MAX.TDF file that indicates the number (and name) of the address registers is in the VARIABLE section. The line is below (note that the keyword DFF after the colon indicates that the "objects" declared by the name Addr_reg[31..0] are D-type flipflops):

```
Addr_reg[31..0]      : DFF;          -- Register the AD[]
```

The line of AHDL responsible for naming and numbering the data signals is a few lines below the address register line:

```
Data_reg[31..0]      : DFF;
```

The lines of AHDL that state the number of tri-state buffers associated with the address and data pins are in the same section (Variable). These lines are listed below (note that the keyword TRI after the colon indicates that the "objects" declared by the names AD_tri[31..0] and Data_tri[31..0] are tri-state buffers):

```
AD_tri[31..0]        : TRI;
Data_tri[31..0]      : TRI;
```

Including/Excluding a Parity Check Function

The capability to check parity, produce a parity signal and produce a parity error signal exist within the AHDL designs for both the Master and Target Interface. Parity is produced via another macrofunction, called pci_par, which is referenced within the Master and Target interface designs (in Altera terminology, the use of lower-level macrofunctions within higher-level macrofunctions is called "instantiation").

Exclusion of the parity check signal and/or parity error signal involves "commenting out" portions of AHDL code (commenting a line out is generally preferable to outright deletion for reasons of ease for future modification, but deletion is an option as well). The lines of AHDL to be commented out correspond to:

(1) The parity and/or parity error pins

(2) The registers and node that hold the parity and/or parity error signals and their output enables

(3) The logic and connections for the parity and/or parity error signals

The declaration of the parity and parity error pins is included in the Subdesign section of the design file. In the MAX_TAR.TDF file, they appear like this:

```
PAR          : BIDIR;          -- Parity
PERR         : BIDIR;          -- Parity Error
```

After being commented out, these lines would appear like this:

```
%    PAR        : BIDIR;        -- Parity
     PERR       : BIDIR;        -- Parity Error     %
```

The registers for the parity and parity error signals (and their output enables) are declared in the Variable section. They appear like this:

```
     PERR_reg   : DFF;
     PERRoe     : DFF;

     PAR_reg    : DFF;
     PARoe      : DFF;

     Par_flag1  : DFF;
     Par_flag2  : DFF;
     Parity     : NODE;
```

The above signals can be commented out by placing a percent sign before the first line and a second percent sign after the last. Finally, the logic and connections for the parity and parity error signals appear in the main body of the design file; percent signs can be used to comment them out in the same manner described above. The signals to be commented out are shown below.

```
     PCI_parity.(AD[31..0], C_BE[3..0]) = (AD[31..0],
     C_BE[3..0]);
     Parity = PCI_parity.(Parity);

     PAR = TRI(PAR_reg, TRDYoe);

     PAR_reg.clk     =    CLK;
     PAR_reg.clrn    =    RST;
     PAR_reg         =    Read_BAR & Parity
                          # !Read_BAR & PAR_reg;

     PARoe.clk       =    CLK;
     PARoe.clrn      =    RST;
     PARoe           =    ADoe;

     PERR            =    TRI(!PERR_reg, PERRoe);

     PERR_reg.clk    =    CLK;
     PERR_reg.clrn   =    RST;
     PERR_reg        =    Par_flag1 & Parity;

     Par_flag1.clk   =    CLK;
     Par_flag1.clrn  =    RST;
     Par_flag1       =    S_data & !Rd_Wr & !IRDY & !TRDY
                          # Write_BAR & !RD_WR & !IRDY &
                          !TRDY;

     Par_flag2.clk   =    CLK;
     Par_flag2.clrn  =    RST;
```

```
Par_flag2       =     Par_flag1;

PERRoe.clk      =     CLK;
PERRoe.clrn     =     RST;
PERRoe          =     S_data & !Rd_Wr & !IRDY & !TRDY
                      # Backoff & !Rd_Wr
                      # Turn_ar & !Rd_Wr
                      # Idle & Par_flag2;
```

A designer who wishes to implement the parity check but not the parity error, can comment out only the AHDL code corresponding to the parity error signal, and this will produce the desired result.

Other Customizations

There are a number of other ways for a designer to modify these macrofunctions. Any number of signals might also be generated for the requirements of the back-end function. Modification of the AHDL code to include the logic equations for these signals is all that is required to implement these signals. Another possible change is to vary the amount of Configuration Space inside the programmable logic device itself. The TAR_MAX.TDF design includes a 27-bit wide register for the BAR. Less registers might be used if the memory requirements did not require the full 27-bit range. A designer might also wish to include more of the Configuration Space inside the programmable logic device, for example the Command or Status Registers. Including more of the Configuration Space inside the programmable logic device is particularly suited to devices that have on-board RAM (such as the FLASHlogic family).

Hardware Implementation

This section discusses the actual implementation of a PCI interface in a programmable logic device. The example that will be used is the Target interface placed into a MAX 7000 EPM7160E device. By understanding how the Target interface fits into the EPM7160E, designers can get a clearer idea of the capabilties of programmable logic in PCI interface applications.

The design file TAR_MAX.TDF was submitted to MAX+PLUS II and compiled, with MAX 7000 as the target family. The design's major features are listed below; a complete listing of the TAR_MAX.TDF design file is available from a number of sources listed at the end of this paper

(1) PCI Target interface with 32-bit address/data connection to PCI bus
(2) 8-bit address and 32-bit data bus to back-end function
(3) Generates parity and parity error signals
(4) Generates system error signal
(5) Includes 27-bit Base Address Register

MAX+PLUS II placed the design into the smallest possible device in the family that would accommodate the design: an EPM7160E in the 160-pin QFP package. The following excerpt from the report file (produced by MAX+PLUS II during compilation) indicates some of the resource utilization:

```
Total dedicated input pins used:    4   /   4 (100%)
Total I/O pins used:                94  / 100 ( 94%)
Total logic cells used:             153 / 160 ( 95%)

Total input pins required:          12
Total output pins required:         17
Total bidirectional pins required: 69
Total logic cells required:         153
Total flipflops required:           123
```

As indicated in the report file excerpt, the design used all of the four dedicated input pins, 94% of the 100 I/O pins and, and 95% of the 160 macrocells. The reamining device resources are available for other functions. Placed into the 12-ns version of the EPM7160E, the design also meets the 33-MHz performance requirement for open PCI systems.

Many PCI Target designs do not require all of the functionality provided by the TAR_MAX.TDF design. For example, some PCI interfaces might require fewer registers in the BAR, or no parity or system error signal generation. Below, Table 1 lists some of these optional functions and the macrocell resources they require; removing these functions (in the case that they were not required) would free up a corresponding amount of resources.

Table 1

Function	**Macrocells Used**
Parity Check	4
Parity Error	4
Base Address Registers	1 per register

If extra resources are required, a designer also has the option to choose a larger device. Two other members of the MAX 7000 family are larger than the EPM7160E: The 192-macrocell EPM7192E and the 256-macrocell EPM7256E. The same PCI Target interface design placed in these devices would yield more extra resources (53 macrocells in the EPM7192E and 103 macrocells in the EPM7256E). The FLEX 8000 devices are also an option; this target interface design occupies about 65% of the resources of the 4,000-gate EPF8452A, leaving about 150 registers and associated logic for other functionality.

Conclusion

A programmable logic solution to a PCI interface offers flexibility and options that a dedicated chip set cannot. These options include the ability to customize the interaction with the back-end design, include or exclude functions that may or may not be needed, and include back-end function logic into the programmable logic device to conserve board real estate. Altera's PCI interface macrofunctions are designed to serve engineers as foundations upon which to build their own PCI interfaces. A number of Altera's devices are suitable for impementing PCI interface designs in addition to the one discussed in this paper, including several members of the MAX 7000, FLEX 8000, MAX 9000, and FLASHlogic families. Finally, Altera's Applications group is available at (800) 800-EPLD to assist any engineer in utilizing the macrofunctions to their best potential.

Obtaining the Macrofunctions

The PCI macrofunctions are available from several sources, including:

(1) Altera's PCI Design Kit (obtainable from Altera Marketing at (408) 894-7000)
(2) Altera's Applications group at (800) 800-EPLD or (408) 894-7000
(3) Altera Applications BBS at (408) 954-0104 in the form of the file PCI_10.EXE
(4) Altera's ftp site: ftp.altera.com

PCI TO VME: BUILDING THE BRIDGE

D. Lisk, T.P. Wilson, A. Sheedy, J. Morris, I. Dobson
Newbridge Microsystems, Kanata, Ontario, K2K 2M5

ABSTRACT

The Peripheral Component Interconnect (PCI) bus has emerged as a popular cure for the local bus bottleneck developing between CPUs and bandwidth-hungry peripherals. Data-intensive real time systems can particularly benefit from the increases in response time allowed by a high performance local bus. Consider also that PCI is welcomed by the likes of IBM, DEC and Apple, and we can predict with some confidence that a PCI interface will be tacked onto a number of high-end peripheral devices. This article addresses some of the technical issues encountered when interfacing PCI with the VMEbus. We discuss maximizing bus performance, cycle mapping (which includes translation of command information, byte lane enabling, endian conversion and address mapping), VMEbus interrupt handling, and the architectural impacts on real time operating systems.

WHY PCI TO VME?

Here's a challenge for you. Pick up a trade journal in computing design or embedded systems and try not to find a mention of PCI (Peripheral Component Interconnect), a hot new local bus sweeping the PC industry. It won't be easy, and indicates the industry-wide acceptance of PCI that promises high volume, low cost PCI peripherals at the right price/performance level. Although originally destined for the PC market, PCI is migrating to industrial applications in the guise of a mezzanine card specification (PMC, IEEE 1386.1). This is where VMEbus enters stage left: an industrial, open standard backplane, inexpensive because of its ubiquity but in need of a local bus partner to take it into the next couple decades.

Just as the VMEbus industry can take advantage of the new spectrum of high end, low cost PCI peripherals, a PCI/VME bridge provides PCI device vendors a window to a large and diverse embedded systems market. Although there have been some rumblings of an industrial strength PCI (ISPCI) rising to compete with VMEbus, there is more incentive at this time to merge the two into one system.

Intriguing technical issues arise when you bridge two such different buses as PCI and VME. The differences go beyond just bandwidth (in 64-bit mode PCI heats up the data bus at over three times the theoretical limit of VMEbus). Fundamental differences also exist in addressing, cycle protocols and interrupt management. One approach in bridging mismatched buses is a decoupled architecture, where the operation of one bus is kept isolated from the operation of the other. This approach is particularly important in bridging PCI and VME, and we'll show that some transactions can only be directly communicated across the PCI/VME bridge by decoupling the buses. Since the following discussion of a PCI/VME bridge focuses on decoupled architecture, it's worth spending page space on a general comparison between coupled and decoupled bridges.

COUPLED VERSUS DECOUPLED

At the risk of over-simplification, you might view a coupled system as a fire brigade of two men and one bucket. One fire-fighter brings water from the river in the bucket and passes it to the second fellow who rushes to splash the water on a burning barn. The bandwidth (flow of water) is only as good as the lowest performance bus, and the first bus must wait while the transaction completes on the second bus (the first firefighter waits for the return of the bucket). Beyond the loss of local bus performance and increase in local bus latency this incurs, there is also the issue of latency constraints on the PCI bus. After the first data beat, the PCI bus will only wait for a limited period before timing out.

In a decoupled system, the fire brigade has two buckets and a big washtub at the halfway point. The first firefighter brings water from the river in his bucket, dumps it in the washtub and rushes back for more. Meanwhile, the second fellow monitors the washtub and as soon as water appears, he uses his bucket to transport water to the fire. The washtub is akin to a decoupling FIFO that passes data from the first bus to the second bus. Since there's no waiting for remote bus arbitration and data acknowledgment, each bus operates at its optimum performance level.

The fire brigade analogy addresses the advantages of decoupling in terms of bandwidth utilization and decreased local bus latency, but masks several technical issues that arise with mismatched protocols in a PCI/VME bridge. These issues revolve around cycle mapping and interrupt translation. In addition, there are considerations for system architects wishing to use a decoupled PCI/VME bridge in their real-time systems.

CYCLE MAPPING

In mapping PCI to VME, keep in mind one over-riding goal. Ideally the PCI bus should be bridged to VMEbus without having to mimic VME. Just because a PCI initiator is sending a transaction to the VMEbus doesn't mean that the PCI bus should suffer from limitations of the VMEbus. The following discussion points out differences between the buses that introduce obstacles to reaching this goal, but also presents ways to surmount these obstacles through decoupled architecture.

Addressing

The two specifications take a fundamentally different approach to address mapping. In PCI, addresses are divided functionally into three separate spaces: Memory, I/O and Configuration. VMEbus on the other hand has a plethora of available address spaces, mostly differentiated according to address width (A16, A24, A32, A40 and A64), although the VME64 specification introduced a configuration space (CR/CSR). To guarantee that a particular PCI address space can be accessed through the VMEbus, you must incorporate the type of PCI access in the VME transaction. This problem is solved by using a PCI/VME bridge with PCI and VMEbus slave images that can be configured to symmetrically map to certain address spaces on the PCI buses.

Beyond the problem of different types of address spaces, VMEbus and PCI have an even more fundamental difference that results from their different evolutionary histories. PCI was spawned in the Intel world making it little-endian, while VMEbus inherited Motorola's big-endian 68K lineage. These different endian systems can be mapped either through Data Invariance or Address Invariance. Using Data Invariance, the PCI/VME bridge would perform operations on primitive data structures (like 8-bit, 16-bit, or 32-bit integers) so that either endian-type processor could read the structure without byte swapping. Unfortunately, the problem with this approach is that Data Invariance scrambles other data structures. The scrambling depends upon the word width used for mapping, and may also depend upon the bus width the data traveled through. A Data Invariant scheme would require software to unscramble the data structures at the destination.

A better approach with a PCI/VME bridge is Address Invariant mapping, which preserves the byte addresses between endian systems. With Address Invariance, the PCI/VME bridge doesn't worry about word width and data structure, but just manipulates byte lane mapping so that the byte ordering appears the same in memory when it goes to either endian system. Note that with this system it is up to software to know the endian interpretation of each multi-byte data field and initiate the required cross addressing or byte swapping.

Command Information

More problems arise in trying to transparently communicate PCI command information across the VMEbus. For example, there are no VMEbus equivalents to PCI's cache line transactions: multiple memory read, memory read line, and memory write invalidate These cache line transactions are meant to optimize bus performance for bulk sequential data transfers. As with the address space problem above, a possible solution would be allowing the PCI/VME bridge to tag VMEbus slave images for certain transactions. For example, a particular VMEbus slave image in the PCI/VME bridge could be programmed to always generate cache line transactions on the local bus. Note that with this solution the system architect must assure that these cache line dependent VMEbus slave images ONLY receive cache line accesses.

Byte Lane Translation

The major difference between PCI and VMEbus in byte lane usage is that PCI follows a much more flexible scheme than VME. VMEbus has a specific and limited set of byte lane "patterns" it can use, while PCI can freely enable or disable byte lanes as required. This difference works in our favor for VMEbus to PCI transactions; the PCI/VME bridge just matches the byte lanes on the PCI bus to the active byte lanes on the VMEbus. In a decoupled architecture, you have the extra advantage of using the decoupling FIFO to pack the VMEbus cycles to the full (64 bit) width of the PCI bus (this is like the second firefighter using a bigger bucket to fetch water from the washtub).

Going the other way (from PCI to VME) raises complications. The byte lane pattern in a particular PCI transaction might not fit with the limited set of available VME cycle types. Under these circumstances, what could be a single transaction on the PCI bus requires multiple cycles on the VMEbus (see Figure 1). In a coupled system, local logic on the PCI board has to abide by VME rules and send its data out to the VMEbus such that it fits with VME cycle types, resulting in loss of PCI bandwidth. However, in a decoupled system the PCI initiator can write its single transaction to the FIFO, receive its handshake, and let the PCI/VME bridge sort out how the transaction should be transmitted on the VMEbus.

Locked Transactions

On the VMEbus, there are two variations of exclusive access. A Read-Modify-Write (RMW) is really an exclusive (indivisible) access to a resource for a read and subsequent write cycle. Note that the locked nature of an RMW is not apparent until the read is completed and the write is begun on the VMEbus. To expand the capabilities of the VMEbus, the VME64 specification introduced

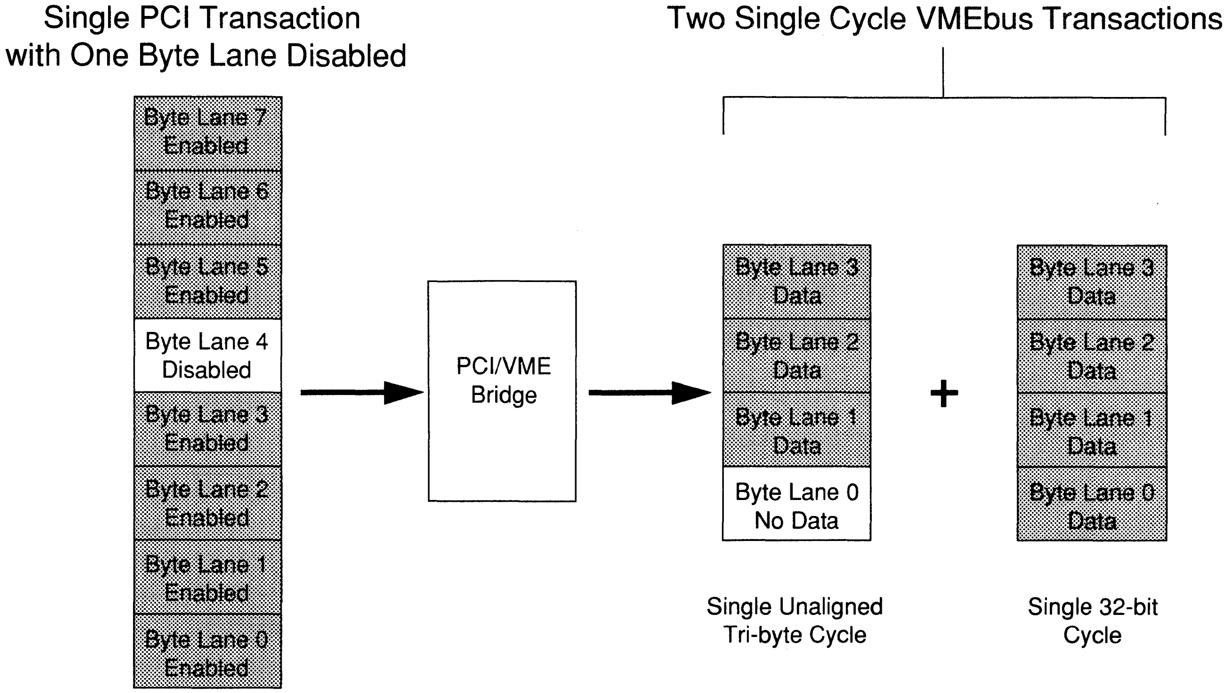


Figure 1 PCI to VMEbus Byte Lane Translation

a locking protocol using the Address-Only-Handshake (ADOH) cycle. After locking a resource, the VMEbus master has exclusive access to that resource for any number of reads and writes in whatever order while it maintains VMEbus mastership.

On the PCI bus, a locked transaction occurs when a local initiator asserts the LOCK# signal. When a resource is locked, the initiator has exclusive read/write access to that resource independent of bus tenure.

Translating a RMW transaction from the VMEbus to the PCI bus presents some difficulties because PCI has no transaction directly corresponding to a RMW cycle. However, the PCI/VME bridge must ensure that a RMW cycle maintains its indivisibility on the local bus. To ensure RMW mapping, the PCI/VME bridge needs to be programmed to honor RMW cycles. This means that a particular VMEbus slave image would generate locked cycles on the PCI bus any time it was accessed by a VMEbus read cycle (the PCI/VME bridge could be similarly programmed to respond to VMEbus lock cycles with a lock cycle on the PCI bus). This guarantees the PCI/VME bridge exclusive access to the local target during the RMW transaction. However, treating every incoming read as a locked access can cause some performance loss on the local bus, and the PCI/VME bridge may hog the LOCK# signal. For these reasons, honoring RMW cycles should be made a programmable slave image option in the PCI/VME bridge.

Translating PCI locked transactions to the VMEbus presents more of a problem because of the different effect bus ownership has on exclusive access on the two buses. Let's work through the chain of events. A local initiator obtains exclusive access to a VMEbus resource by locking the local bus and then locking the VMEbus. Since the PCI initiator assumes that it has exclusive access to the resource over multiple transactions, the PCI/VME bridge must not release the VMEbus after each PCI transaction. If the PCI/VME bridge releases the VMEbus, then it loses exclusive access to the VMEbus resource. So how does the PCI/VME bridge know when the local processor is finished with its exclusive access?

At first glance, you might think that end of the local lock would be a reasonable indication that the exclusive access is complete. However, the LOCK# signal is not in general use in all PCI systems, so a more general solution would be desirable. One way to override the bridge's requirement to release the VMEbus after each PCI transaction is by allowing the PCI/VME bridge to park on the VMEbus. This might be implemented through a VMEbus ownership toggle that could be set by local logic. It is then up to local logic to tell the PCI/VME bridge when the local initiator is finished with its exclusive access to the VMEbus resource.

Block (Burst) Transactions

A block (or burst) transaction transfers large amounts of data by issuing several data beats within a single transaction. A major hurdle in translating block transfers between the VMEbus and PCI is PCI's latency requirements. For instance, with block writes from the PCI bus to the VMEbus, the PCI device cannot wait for data acknowledgment from the remote slave and the transaction times out. Likewise with block writes from the VMEbus to the PCI, the time required for handshakes to propagate from the remote bus causes the PCI bus to time out between data beats. In a coupled system, the PCI/VME bridge must always break block writes from one bus into single cycles on the destination bus. This creates a significant degradation in performance, an ironic result since block transfers are designed to improve data transfer rates.

The only way to directly translate block writes between the PCI bus and the VMEbus is through decoupling. With a decoupled architecture, block writes can be loaded into one end of a FIFO at the optimum data transfer rate for that bus. The data can then be unloaded as block transfers to the destination bus from the other end of the FIFO.

A slightly different approach is used for block reads from the VMEbus (VME master reading from a PCI resource). In a coupled system, the PCI resource provides the first block and waits for the VMEbus master to ask for more (the second beat). However, the VME master is unable to respond quickly enough and PCI latency constraints will time out the transaction. The only way a VME master can perform its block read is if the PCI/VME bridge pre-fetches. Pre-fetching means that when the PCI/VME bridge receives a block read request from a VME master, it fills its FIFO from the local resource using local block reads. The VME master can then perform block reads on the VMEbus from the other end of the FIFO. To return to the fire brigade, imagine one fire-fighter sees a burning barn and tells the other fellow. The second fire-fighter then rushes back and forth between the wash tub and the river until the wash tub is full. If the entire contents of the washtub is used on the fire and the first fire-fighter says the barn is still burning, then the other fellow fills the tub again.

That takes care of block reads from the VMEbus to the PCI bus, but how about the other direction (PCI master reading from a VME resource)? In a coupled system, PCI latency requirements will make the PCI/VME bridge terminate the local transaction after every data beat because the VME resource cannot respond quickly enough. Unfortunately, pre-fetching won't help under these circumstances because even with block transfers the VMEbus cannot load a FIFO quickly enough to keep up with the data transfer rate on the PCI bus. The bottom line is that block reads from the PCI bus to the VMEbus will always be broken into single read cycles.

Interrupts

Interrupt mapping presents special problems because the VMEbus provides a much richer interrupt environment than PCI. As compared to the seven available interrupt levels per module on the VMEbus, PCI bus allows for only one interrupt per single function device. The interrupt acknowledgment (IACK) cycles on the two buses are also fundamentally different. The VMEbus has a 68K-like interrupt acknowledgment cycle while PCI follows an interrupt control protocol from the PC world.

Interrupts translated from the VMEbus to PCI cannot be coupled to the PCI interrupt controller because the coupled read to obtain the VMEbus interrupt vector does not fit within teh timing requirements of the PCI interrupt cycle. One way around this is to decouple the interrupter from the interrupt handler. When the PCI/VME bridge receives an appropriate VMEbus interrupt, it immediately acknowledges the VMEbus interrupter and obtains the interrupt vector. Once the interrupt vector is obtained, the PCI/VME bridge relays the interrupt to the PCI interrupt controller.

Timing constraints also keep the PCI/VME bridge from coupling local interrupts to the VMEbus. It is not possible for the PCI/VME bridge to obtain interrupt acknowledgment from the VMEbus interrupt handler quickly enough to meet the timing requirements of the PCI interrupt cycle. Any locally initiated interrupts relayed to the VMEbus will require the PCI/VME bridge to supply an interrupt vector.

DECOUPLING IMPACT ON REAL-TIME SYSTEMS

Earlier, we showed that decoupling is the only means to directly translate block transfers between PCI and VMEbus. By real-time, we mean a system with a large number of independent tasks, each with their own particular deadline. The success of the system depends not only on tasks being performed, but also on whether the task is completed before its deadline. This means that timing constraints must be guaranteed in a real-time system. These timing constraints are affected by three parameters: bandwidth, latency, and priority. Beyond just getting the job done in time, the tasks must also be completed within a certain fault tolerance.

Bandwidth and Latency

Completing a particular task before its deadline depends upon how quickly previous tasks are performed. Decoupling makes better use of bandwidth by allowing the CPU to perform some tasks in parallel, something not possible in a coupled architecture. A better use of bandwidth decreases latency and increases system performance. Figure 2 below illustrates how this works. In a decoupled system, a CPU on the PCI bus can write data to a decoupling FIFO and then perform vari-

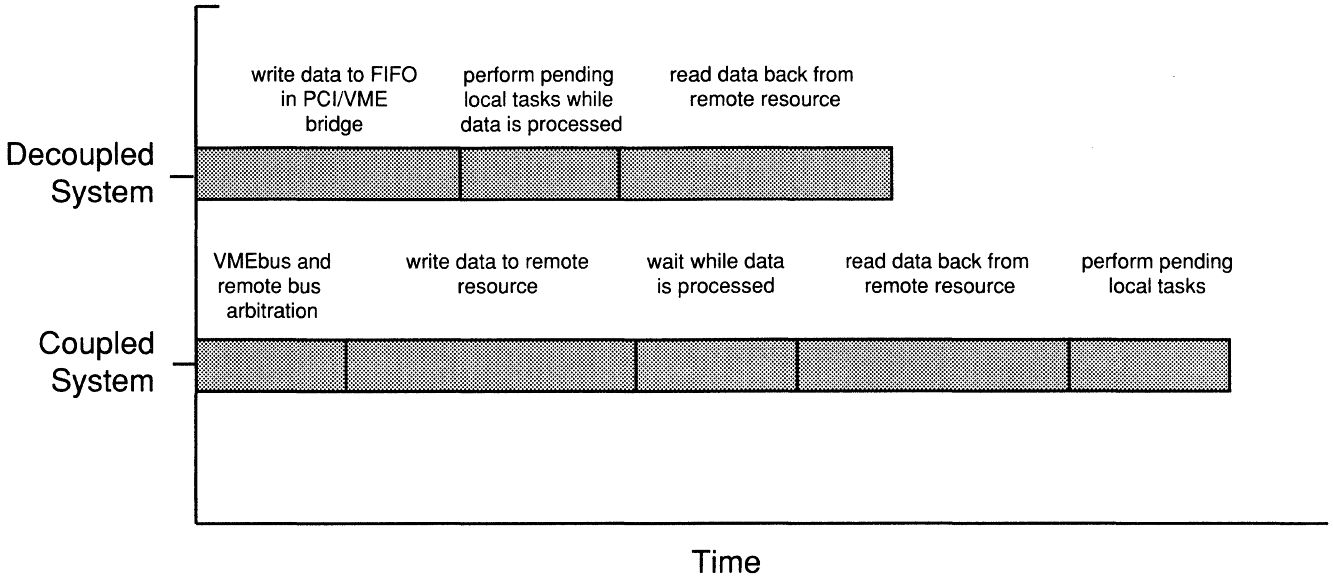


Figure 2 Parallel Task Performance with Decoupling

ous local tasks while the PCI/VME bridge sends the data to the remote resource across the VMEbus. After a certain programmed period (perhaps an interrupt is sent back to indicate processing is complete), the CPU reads the processed data back from the remote bus. In a coupled system, the CPU would write the data to the remote resource, wait while the data was processed, and then read the results back (it's better to have the CPU wait than re-arbitrate the entire coupled link twice). Only after this coupled procedure is completed can the CPU perform the pending local tasks.

An argument could be made that increases in bandwidth utilization through decoupling incur some cost in latency. For example, a local processor queues a transaction in a decoupling FIFO, but the queued transaction must wait while entries closer to the front of the FIFO are processed. However, this FIFO latency is compensated for by the decrease in local bus latency brought about by decoupling. To make this point, let's say an I/O port on the PCI bus receives data that must be transferred to its destination across the VMEbus within a certain deadline (due to FIFO constraints in the I/O device). The I/O device cannot wait too long while the local processor is tied up arbitrating access to a remote local bus or the older I/O data will be over-written. In a decoupled system, the processor writes the I/O data to the decoupling FIFO with a minimum latency. What little latency there is in the FIFO is probably less important than the immediate servicing of local tasks (low local bus latency).

Along these same lines, real-time system architects are often concerned with low interrupt latency on the local bus. If a processor in a coupled system is tied up with accessing remote buses, then there may be inordinate delays in servicing local interrupts. A decoupled system guarantees system designers a lower interrupt latency on the local bus.

Priority

Prioritization in a real-time system is an issue when several tasks compete for bandwidth. In some instances, a deadline for one task will slip (experience inappropriate latency) so that a higher priority task meets its correct timing requirements. For example, system control in a battle ship might need to decide between activating the laundry mechanism in time for Thursday's wash or responding to incoming missiles. Regardless of the ripeness of the laundry, the priority of the two tasks is clearly defined in the system.

Priority inversion can occur in a system where a high priority task (that pesky missile attack) is queued behind a low priority task (Thursday's laundry). As discussed above in Bandwidth and Latency, transactions are completed with less latency in a high performance decoupled system than in a high latency coupled system. There is some latency introduced in a decoupled system while a FIFO entry works its way to the head of the queue, and the system architect will need to allow for worst case FIFO latency in designing the decoupled system.

Fault Tolerance

One other crucial aspect of a real-time system is fault tolerance. All tasks have some degree of fault tolerance (think again about that laundry/missile comparison). Fault tolerance is divided into fault detection, fault isolation, and fault recovery. Decoupling doesn't really affect the first two, but does impact fault recovery. Let's revisit the fire brigade analogy. Even though it is known that a bucket of water is dropped by a particular fire fighter (fault detection and fault isolation), the decoupled system makes it difficult to ask for a replacement (fault recovery). In a real-time system, by the time the error is detected and isolated, the origin for the data may be over-written or the deadline for the task may have passed.

The fault recovery mechanism employed depends on the type of error. For example, if there's a bus error and only command information is lost, then the PCI/VME bridge can retry the transaction from the data in the original FIFO entry. If there's an error where the data received at the far end is mostly corrupted, then there's more of a problem. One way around this problem in a decoupled system would be to buffer past transactions as insurance against errors. For example, you might want to buffer I/O data in case there is a transmission problem across the VMEbus. You can also use barrier transactions to check on previous transactions. A system might read from a location that has just received data to ensure that the correct information is present.

THE BRIDGE IS BUILT

This discussion of a PCI/VME bridge is not completely rhetorical, and indeed comes from practical silicon design experience. Newbridge Microsystems has already used a decoupled approach to construct such a PCI/VME bridge: humbly dubbed the Universe™. The Universe™ provides the PCI/VMEbus interface that will allow embedded systems designers to bring the new wave of high end components with a PCI interface to their open VMEbus systems (Andrews, 1994a). This is especially important considering the modular approach VMEbus board designers are taking with their PCI products, where boards can be assembled from PMC modules with a standard (PCI) interface (Andrews, 1994b). Beyond just having processor/DRAM modules and PCI-based I/O modules at their disposal, board designers can now rely on an off-the-shelf PCI/VME bridge that will bring all this PCI local bus power to the VMEbus.

REFERENCES

Andrews, W. 1994a. "VME bridge chip clears way for industrial PCI," Computer Design, July 1994 , pp 44-47.

Andrews, W. 1994b. "VME gets facelift with new-generation processor," Computer Design, July 1994 , pp 38-40.

Designing A PCMCIA Add-In Card for the PCI Bus

By Allen M. Light
Technical Marketing Engineer
Intel Corporation
1900 Prairie City Rd.
Folsom, CA 95630
Ph. (916) 356-4486

The PCI bus, a 32- or 64-bit interface designed for use in PC systems and performance peripherals, is regarded by many OEMs and system developers as the leading edge in affordable, performance local buses. Built with peripheral compatibility, processor independence and upgradeability as key requirements, PCI protects end user investments in technology and interfaces. For these reasons and more, the PCI bus provides a relatively easy interface for most add-in cards. This article describes how a PCMCIA controller -- the 82092AA -- was designed to enable a PCMCIA add-in card to interface with the PCI local bus. The process for designing controllers for other add-in cards is nearly identical.

Why build a PCI add-in card? There are many viable reasons. First, industry analysts estimate that PCI will ship in almost 60 million PC systems by the end of 1997, a six-fold increase over the estimated 10 million systems sold by the end of this year (Dataquest and In-Stat figures). Second, PCI delivers the advantages of superior power, performance, plug-and-play compatibility and price necessary for todays and tommorow's PC systems. By way of introduction to the PCI local bus, some of these advantages are briefly detailed below.

About the PCI bus

Most PC's implemement PCI as a 32-bit, 33MHz local bus capable of throughput rates of 133MB per second in burst mode. Although PCI's performance is similar to that of a direct connection to the processor local bus, it is in fact physically removed from the processor by a PCI bridge. This PCI bridge provides a managing layer between the CPU and peripherals and presents a uniform interface that provides streamlined, efficient data transfer. It also provides support for bus mastering, which enables intelligent devices to directly access main memory. PCI also includes an optional burst mode that enables accelerated throughput of data across the bus. Peripherals are synchronized to the PCI clock, which is typically based on the microprocessor and its support circuitry. The PCI local bus specification, revision 2.0, defines an operating clock rate ranging from DC to 33MHz.

The key to a successful PCI add-in card design is simply a good understanding of the PCI 2.0 specification. By carefully following the elements in the specification, developers are assured of building performance, high value add-in cards for PC systems. Among the benefits of using PCI is auto-configuration which eliminates the need to set switches or jumpers.

Finally, there are several inherent features of PCI contribute to the low cost of the system. These include multiplexed pins for lower pin count and packaging costs; the elimination of high-powered bus drivers; and glue-less connections. This simplifies the controller design.

Although it performs like a direct connection to the processor bus, the PCI bus is, in fact, physically removed from the processor by a bridge (see Figure 1). The bridge serves as a managing layer between the CPU and peripherals, presenting a uniform interface that streamlines data transfers. The bridge also gives intelligent devices (serving as bus masters) direct access to main memory.

Peripheral devices are synchronized to the PCI clock, which is typically tied to the microprocessor and its support circuits. Revision 2.0 of the PCI Local Bus Specification calls for a clock frequency up to 33 MHz. The specification also defines two types of peripherals: targets and—intiators (formally, master / slave). A target can receive or send data, but only under the control of a master (initator), which can drive address, data and control signals on the PCI bus.

Although PCI is described as a "glueless" interface, this characterization needs clarification. Logic is needed to connect a peripheral to the PCI bus. Where the interface logic can be integrated into the peripheral device, the connection is effectively glueless. Where it cannot, a separate controller chip is needed. In this example, we used a 208 pin ASIC device as our PCMCIA controller.

A typical target transaction starts when the PCI controller decodes its address by comparing the PCI address against the target base address register. In response, the controller sends address bus and byte enable signals and, for write operations, data to the back-end device. It also issues read/write control and configuration access signals to tell the back-end device what kind of transaction is taking place, as well as a device acknowledge signal, which indicates that all signals are being driven with valid values.

When the back-end I/O device finishes the transaction, it asserts a device request signal, saying that it has finished the transaction, while at the same time issuing the requested data for read transactions. When the controller finishes the PCI transfer, it de-asserts a device acknowledge line on the back-end I/O. At that time, the back-end device de-asserts its request signal or, if warranted, sends an error (abort or retry) signal to the controller.

Adapter Operation

Rev 2.0 of the PCI specification defines several types of bus commands.

Among them are memory, I/O, and configuration read and write cycles. The different cycles are identified by the *Bus Command* and *Byte Enable* pins (C/BE[3::0]#). These commands provide compatibility for the typical memory and I/O bus commands on X86 architecture machines as well as additional commands for performance and configuration.

A generic read transaction (see Figure 2) begins when the master asserts *Cycle Frame* (FRAME#), then a valid address AD[31::00] and valid bus command C/BE[3::0]#. The target device will then assert *Device Select* (DEVSEL#) and have access to the shared address and data bus AD[31::00]. The data is transferred and the cycle is complete when the both the *Initiator Ready* (IRDY#) and *Target Ready* (TRDY#) are asserted.

A generic write transaction (see Figure 3) is similar to a read operation except no turnaround cycle is required following the address phase. In this case the master provides both address and data. The data phase works the same for both read and write transactions.

PCI-based PCMCIA Controller Architecture

To start designers on the way to building a custom peripheral interface to the PCI bus, the PCI System Design Guide suggests a generic I/O controller architecture consisting of PCI interface logic, control logic for the "back-end" I/O function, configuration register space and optional expansion ROM (see Figure 4). Elements such as buffers (including FIFO buffers) and registers comprise the device-specific function of the controller.

All signals between the back-end I/O device and the PCI interface are synchronous. In this way, the controller can accommodate the widest range of device speeds and prevent data contention on bi-directional signal wires. Certain signals and functions within the controller distinguish target from master devices, but the basic architecture is the same for both. In the case of the 82092AA PCMCIA controller, it is a target (slave) device.

Since the 82092AA is a multi-function PCI device, it connects the PCI bus to several back-end devices. In this case there are five back-end (four PCMCIA sockets and one IDE controller) devices bridged to the PCI bus via the 82092AA (see Figure 5). This allows for several expansion options for PCMCIA and IDE while physically only taking up one PCI slot on the system (or one PCI load if implemented on the mother board).

This approach has several advantages over ISA based PCMCIA and IDE bridges. For the IDE controller, the 82092AA provides fully programmable timings providing support for mode 0 through mode 3 IDE devices. Standard ISA IDE controllers are not capable of supporting the faster cycle time required by the new enhanced IDE specification. Similarly, the PCMCIA specification supports memory reads and writes with 80ns window timings. The PCI bus has the bandwidth to support these timings where the ISA based devices stop at 250ns window timings.

Pin Descriptions

The 82092AA is a 208-pin component. Of the 124 pins on a 32-bit PCI connector, only 47 are essential for a target device; only 49 for a master. These pins comprise the address, data and control functions and are identical between target and master, except for the Request and Grant arbitration lines that are exclusive to a master.

The essential signals are divided into four types. The first two types are conventional input-only and tri-state (high, low and high-impedance) I/O signals commonly found on programmable chips. The remaining two, however, are another matter. Of these, one is called a sustained tri-state signal and must be actively de-asserted -- pulled high, since it is active low -- for one cycle before it switches to a high-impedance state.

Active de-assertion serves two purposes; it improves system performance by relinquishing the line in at most one cycle, and it saves power by minimizing the time the CMOS bus floats. Pull-up resistors on the system board sustain the de-asserted state until the signal line is again actively pulled low. The fourth pin type is an open drain, which allows several devices to be connected in a wire-OR configuration. Of the essential signals, only system error, SERR#, requires an open drain pin.

The efficiency of the PCI interface is evident when you compare the 47 pins on the PCI interface (the 32 address, data, and control signals are multiplexed) to the 85 pins used for the PCMCIA interface.

Reflective Wave Technology

Because the PCI bus is an unterminated transmission medium with CMOS loads, its steady state current is very low. As an unterminated bus, PCI's signaling technology takes advantage of reflected and incident waves to maintain strong signal integrity. Simply put, when an incident wave traveling along the bus reflects off the unterminated end, the reflected wave combines with the incident wave, doubling the effective voltage at the receiver input. Consequently, a PCI driver need only drive the bus to -- ideally -- half the receiver's required high (V_{ih}) or low (V_{il}) switching level.

During the propagation of the wave, the signal level is in the middle of the switching range. Propagation time varies with the electrical length of the bus and can last up to 10 ns (one third of a clock period at 33 MHz). A controller that does not adhere to the V-I specification must reduce its operating frequency to 25 MHz, extending the bus cycle by 10 ns, thereby allowing for an additional reflection to drive the receiver input to its required level.

Due to the reflective nature of the PCI unterminated medium, an ASIC used as a controller cannot be evaluated solely on the basis of I/O buffer DC sink (I_{ol}) or source (I_{oh}) capabilities. When selecting a suitable ASIC, you must verify that the I/O buffers adhere to the V-I specification stated in the Rev 2.0 of the PCI specifiecation. This is crucial to verifiy the signal integrity of the entire system. If the ASIC vendor does not supply V-I characterization or simulation models (SPICE or IBIS), verification can be achieved through curve trace measurements.

The PCI electrical definition allows both 3.3- and 5-Volt (V) logic levels. Although a trend toward 3.3V is underway to reduce power consumption, the predominant interface is still 5V. PCI drive requirements are specified as AC characteristics plotted in V-I curves, rather than as DC drive levels. The curves assume a maximum of 10 loads, where peripherals on the system board count as one load and PCI peripherals in slots count as two. (It should be noted that AC characteristics are shown in the PCI specification as V-I plots, not DC loads.)

Auto-configuration

Autoconfiguration -- the ability of system software to automatically identify and configure add-in cards -- eliminates the configuration switches and jumpers commonly found on ISA cards. Importantly, it takes the technical complexity out of adding expansion cards and makes the PC system much easier to enhance. To accommodate autoconfiguration, the PCI specification calls for PCI agents to include registers for storing configuration information. It also reserves 256 addresses, 64 of which have been defined, for accessing those registers.

As a minimum requirement for supporting autoconfiguration, a PCI component must store its vendor ID, device ID, command register and status registers, revision ID, class code and header type. In addition, a target device that contains addressable memory or I/O space should also have a base address register. Making this register available lets autoconfiguration software move the device's address range and, therefore, avoid conflicting address among multiple targets. In the case of the 82092AA, one I/O base address register is available for the PCMCIA port. This can be set to 3E0h to be compatible with the standard 82365SL ISA PCMCIA controller or to any other I/O address. It also allows for four base addresses for the IDE function. These can be set to the standard IDE addresses for primary and secondary or any other I/O address.

General Considerations

Bus Loading

When designing for the PCI specification, a number of bus loading considerations should be kept in mind. Specifically, it is a violation of the PCI spec to:

- Attach an expansion ROM directly (or via transceivers) to any PCI pins.
- Attach two or more PCI devices on an expansion board, unless they are placed behind a PCI-to-PCI bridge.
- Attach any logic (other that a single PCI device) that looks at PCI pins.
- Use a PCI component set that places more than one load on each PCI pin; e.g., separate address and data path components.
- Use a PCI component that has more than 10pF capacitance per pin.
- Attach any pull-up resistors or other discrete devices to the PCI signals.

Layout Considerations

Transmissions Line effects

To keep a lid on transmission line effects, the PCI specification recommends a pin-out configuration that minimizes signal trace lengths, or "stubs" (see Figure 6). The maximum allowable trace length for 32-bit signals is 1.5 inches. Other signals are limited to two inches, except for the clock signal, which can extend to a maximum of 2.5 inches. The suggested pinout refers to a 132-pin plastic quad flat pack (PQFP) component. Using a device with more package pins, however, can further reduce trace capacitance and inductance if it can be configured with fewer traces wrapping around it.

Load Capacitance

Other important specifications address load capacitance, timing parameters and device protection. These are spelled out fully in the official specification. Briefly, to avoid excess loading, the capacitance of a system board PCI peripheral input pin should not exceed 10 pF on a signal line or 12 pF on a clock line. PCI peripheral card designs (two PCI loads per the specification) should keep the combined capacitive load of the PCI controller device, PCI connector and printed circuit trace below 20pF. Also, because of the electrically reactive nature of the PCI environment, the specification recommends that PCI agents withstand an 11-V overshoot and a 5.5-V undershoot pulse of 11-ns duration.

Conclusion

PCI add-in card design is a simple process, if you have a clear understanding of the PCI specification, and the system approach that was used to develop the spec. By using these tools, and carefully considering the unique aspects of the PCI local bus, developers can build reliable, high performance, high value add-in cards for PCI-based PC systems.

Allen M. Light is a Technical Marketing Engineer in Intel Corporation's PCI Chipset Division.

AUTHOR INDEX

KEYWORD INDEX

PARTICIPANTS LIST

Shrikant Acharya, Margi Systems, Inc., 12350 E. Del Amo Blvd., #1110,
Lakewood, CA 90715-1714, 310-809-4362, Session D1

Barbara Aichinger, FuturePlus Systems Corp., 36 Olde English Rd., Bedford,
NH 03110, 603-471-2734, Tutorial T2B, Session D8B

Dennis Aldridge, Texas Microsystems, P.O. Box 42963, Houston, TX 77242,
713-541-8200, Session D8A

Charles Anderson, Cogent Data Technologies Inc., 175 West St., PO Box 926,
Friday Harbor, WA 98250, 360-378-2929, Session 2B

David Baker, Brooktree Corp., 10802 Chateau Hill, Austin, TX 78750, 512-
502-1701, Session D5B

Stan Baker, PREP Corp., 504 Nino Ave., Los Gatos, CA 95032, 408-356-5119,
Session D4D

Charlie Barbour, FirePower Systems, 190 Independence Dr., Menlo Park, CA
94025, 415-462-3030, Session 5C

Bob Beachler, Altera Corp., 3 W. Plumeria, San Jose, CA, 95134, 408-894-
7187, Session 1C

Jim Beedle, In-Stat, Inc., 7418 E. Helm Dr., Scottsdale, AZ 85260, 602-483-
4463, Session 5A

Michael Benson, Fore Systems, 174 Thorn Hill Rd., Warrendale, PA 15086-
7835, 412-772-8609, Session 2B

Kenneth Birch, Micron Computer Inc., 1316 Shilo Drive, Nampa, ID 83687-
3045, 208-463-3584, Session 2C

John Birkner, QuickLogic, 2933 Bunker Hill Lane, Santa Clara, CA 95054,
408-987-2000, Session D4D

Menachem Blasberg, Corelis, 12607 Hidden Creek Way, Suite H, Cerritos, CA
90701, 310-926-6727, Session 5D

Mitch Bradley, FirmWorks, 480 San Antonio Rd., Mountain View, CA 94040,
415-917-0100, Session 5C

Kimball Brown, Dataquest, 1290 Ridder Park Dr., San Jose, CA 95131, 408-437-8235, Luncheon Presentation

Marshall Brumer, Microsoft, One Microsoft Way, Redmond, WA 98052, 206-936-5840, Session D2

Sean Burke, Dell Computer Corp., 2214 West Breaker Lane, Austin, TX 78758, 512-728-4984, Session 2C

Tom Caldwell, Rockwell Network Systems, 7402 Hollister Ave., Santa Barbara, CA 93117, 805-562-3103, Session 2B

Wen-Chi Chen, VIA Technologies, Inc., 5020 Brandin Ct., Fremont, CA 94538, 510-683-3316, Sessions D3E, 2A

Yu-Ping Cheng, AdvanSys, 1150 Ringwood Ct., San Jose, CA, 95131, 408-383-9400, Session 3B

Larry Chisvin, Western Digital Imaging, 800 E. Middlefield Rd., Mountain View, CA 94043, 415-335-2614, Session 2D

Michele Clarke, EE Times, 300 5th Ave., Waltham, MA 02158, 617-487-7544, Session 1D

Mark Clayton, Ariel Corp., 433 River Road, Highland Park, NJ 08904, 908-249-2900, Session 5B

Donald Coffin, Future Domain Corp., 2801 McGaw Ave., Irvine, CA 92714, 714-253-0508, Session 3B

Frances Cohen, Phoenix Technologies Inc., 2575 McCabe Way, Irvine, CA 92714, 714-440-8323, Session 5A

Vincent Coli, APTIX Corp., 2890 N. First St., San Jose, CA 95134, 408-428-6234, Session 5D

Ken Comstock, Diamond Multimedia Systems, 2880 Junction Ave., San Jose, CA, 95134, 408-325-7000, Sessions 3C, 5C

Steve Cooper, I-Bus, 9596 Chesapeake Dr., San Diego, CA 92123, 619-974-8424, Session 3C

Frank Creede, Logic Innovations, Inc., 6205 Lusk Blvd., San Diego, CA 92121, 619-455-7200, Session 4A, Technology Fair

Claude Cruz, National Semiconductor Corp., 333 Western Av., M/S 10-26, South Portland, ME 04106, 207-775-8318, Session 4D

Mike Culver, Acer America Corp., 2641 Orchard Pkwy., San Jose, CA 95134, 408-432-6200, Session 2C

Alak Deb, Synopsys, 700 E. Middlefield Rd., Mountain View, CA 94043, Session 5D

Alan Deikman, Znyx Corp., 48501 Warm Springs Blvd., Fremont, CA 94539, 510-249-0800, Session 2B

Dave Deming, Solution Technology, PO Box 104, Boulder Creek, CA 95006, 408-338-4285, Session 3B

John Derrick, IBM Microelectronics, 1000 River St., Essex Junction, VT 05452, 802-769-6525, Session D7B

Joseph DiMartino, IBM PC Company, 3039 Cornwallis Rd, MS 201-K103B-91B, Research Triangle Park, NC 37709, 919-543-9795, Session D8A

Boris Elisman, Hewlett-Packard Co., 3000 Hanover St., Palo Alto, CA 94303, 408-857-1501, Session 2C

John Elmore, IBM, 1000 Northwest 51st St., Boca Raton, FL 33431, 407-982-7196, Session D8A

Joe Eschbach, rPC, 215 Moffett Park Dr., Sunnyvale, CA 94089, 408-541-6100, Session 1B

David Evoy, VLSI Technology, 8375 S. River Pkwy., Mail Stop 260, Tempe, AZ 85285, 602-752-6481, Session 2A

David Fair, Digital Semiconductor, 77 Reed Rd., MailStop HLO2-2/M9, Hudson, MA 01749-2895, 508-568-5157, Session 1C

Daniel Faizullabhoy, Adaptec, 691 S. Milpitas Blvd., Milpitas, CA 95035, 408-945-8600, Session3B

Bradly Fawcett, Xilinx Inc., 2100 Logic Dr., San Jose, CA 95124, 408-879-5097, Session D3D

Wayne Fischer, Force Computers Inc., 2001 Logic Dr., San Jose, CA 95124, 408-369-6260, Tutorial T1C, Session 3A

Richard Fisher, Microchip Technology Inc., 2355 West Chandler Blvd., Chandler, AZ 85224, 602-786-7291, Session D10C

Jeffery Floyd, Motorola MDAD Applications, 3510 Ed Bluestein Blvd., Mail Drop K5, Austin, TX 78721, 512-933-7365, Session D8B

Steven Freear, University of Leeds, Dept. of Electrical Engineering, Leeds, West LS2 9JT, United Kingdom, +44-532332016, Session 1D

Martin Freeman, Philips Research, 4005 Miranda Ave., Palo Alto, CA 94304, 415-354-0329, Session D10C

Michael Garcia, Motorola Semiconductor Products Sector, 6502 William Cannon Dr., Austin, TX 78735, 512-795-7198, Session D4C

Billy Garrett, Rambus, Inc., 2465 Latham St., Mountain View, CA 94040, 415-903-3800, Sessions D3B, 1B

Brian Gibson, S3, Inc., 2770 San Tomas Expy., Santa Clara, CA 95051, 408-980-5400, Session 2D

Glen Gibson, AMD, PO Box 3453, Sunnyvale, CA 94088, 408-749-5466, Session 4C

Byron Gillespie, Intel Corp., CH5-233, 5000 W. Chandler, Chandler, AZ 85226, 602-554-2653, Session D5E

Jerry Gipper, Motorola Computer Group, 2900 S. Diablo Way, Mail Stop DW212, Tempe, AZ 85283, 602-438-3025, Session 3A

Randy Giusto, BIS Strategic Decisions, One Longwater Circle, Norwell, MA 02061, 617-982-9500, Session 6A

Ernest Godsey, Interphase, 13800 Senlac, Dallas, TX 75234, 214-919-9000, Session 3A

Gary Griffiths, IBM Power Personal Systems, Route 100, PO Box 100, Bldg. 3, 3L10, Somers, NY 10589, 914-766-3121, Keynote Presentation

David Gustavson, SCIzzL, 1946 Fallen Leaf Lane, Los Altos, CA 94024-7206, 415-961-0305, D10A

Michael Harris, Avance Logic, 46750 Fremont Blvd., Suite 105, Fremont, CA 94538, 510-226-9555, Session D7E

Mike Hawkey, Western Digital Imaging, 800 E. Middlefield Rd., Mountain View, CA 94043, 415-335-2590, Session 1B

Scott Hay, Intel Corp., HF3-64, 5200 N.E. Elam Young Pkwy., Hillsboro, OR 97124, 503-264-2217, Session D6

Peter Hayden, Digital Equipment Corp., 30 Porter Rd., Littleton, MA 01460, 508-486-2437, Session 5A

Tim Hennessey, SystemSoft, 313 Speen St., Natick, MA 01760, 508-651-0088, Session 5A

Greg Hill, FirmWorks, 480 San Antonio Rd., Mountain View, CA 94040, 415-917-6985, Session D5C

Rainer Hoffmann, Thesys Gesellschaft fur Mikroelectronik, Haarbergstrabe 61, Erfurt, Germany, 99097, +361-4278355, Session 5B

Lloyd Holder, NSTL/McGraw-Hill, 625 Ridge Pike, Bldg. D, Conshohocken, PA 19428, 610-941-9600, Session D8B

Bill Holland, IBM Networking Group, Bldg. 002, Dept. C12A, 3039 Cornwallis Rd., Research Triangle Park, NC 27709, 919-543-5444, Session 1D

Dick Holmberg, American Megatrends, 6145-F Northbelt Pkwy., Norcross, GA 30071, 404-246-8669, Session 5A

Phil Hood, NewMedia Magazine, 901 Mariner's Island Blvd., Suite 365, San Mateo, CA 94404, 415-573-5170, Luncheon Presentation

Jim Hora, ZeitNet, Inc., 2255 Martin Ave., Santa Clara, CA 95050, 408-562-1880, Session 2B

Joseph Hustein, GEC Plessey Semiconductors, Inc., 1500 Green Hills Rd., Scotts Valley, CA 95067-0017, 408-439-6073, Luncheon Presentation

Barry Isenstein, Mercury Computer Systems, 199 Riverneck Rd., Chelmsford, MA 01824, 508-256-1300, Sessions D10F, 3A

Dale Jorgensen, Intel, 1900 Prairie City Rd., Folsom, CA 95630, Session 2A

Jim Joseph, Ramtron, Inc., 1850 Ramtron Drive, Colorado Springs, CO 80921, 719-481-7057, Session D5A

Art Kahlich, Solliday Engineering Corp., 1756 18th St., San Francisco, CA 94107, 512-933-6277, Tutorial T2C

Aki Kaniel, Philips Semiconductor, M/S 45, 811 E. Arques Av., Sunnyvale, CA 94088, 408-991-2619, Session D7D

Sean Keohane, National Semiconductor, 2900 Semiconductor Dr., Santa Clara, CA 95052-8090, 408-721-3498, Session 1C

Gary Kidwell, Interphase, 13800 Senlac, Dallas, TX 75234, 214-919-9000, Session 2B

Robert Kilmartin, IBM Microelectronics, 1000 River St., Essex Junction, VT 05452, 802-769-6525, Session 2A

Dave Kresta, Logic Modeling Corp., 19500 N.W. Gibbs Dr., Beaverton, OR 97075, 503-531-2249, Session D8B

Gary Kuzmin, Aavid Thermal Technologies, One Kool Path, Laconia, NH 03247, 603-528-3400, Session1A

Dave Lawrence, Ventura Micro, Inc., 200 South A St., Suite 208, Oxnard, CA 93030, 805-486-6686, Tutorial T1E

Stephen Lawton, Digital News & Review, 2511 Carmel Dr., San Bruno, CA 94066, 415-588-8255, Session 4C

Jeffrey Lee, First International Computer, 980A Mission Ct., Fremont, CA 94539, 510-252-7777, Session 4B

Sheau-Jiung Lee, Acer Laboratories, Inc., 2641 Orchard Pkwy., San Jose, CA 95134, 408-435-5262, Session D7A

Steven Leibson, EDN Magazine, 275 Washington St., Newton, MA 02158, 617-558-4214, Session 1A

Lillian Leung, FirePower Systems, 190 Independence Dr., Menlo Park, CA 94025, 415-462-6217, Session 6A

Markus Levy, EDN, 1936 Sheffield Dr., El Dorado Hills, CA 95762, 916-939-1642, Session 2A

Allen Light, Intel, 1900 Prairie City Rd., Folsom, CA 95630, 916-356-4486, Session 4D

David Limp, Apple Computer Inc., One Infinite Loop, MS:306-4PM, Cupertino, CA 95014, 408-862-7218, Session 6A

Tzu-Mu Lin, VIA Technologies, Inc., 5020 Brandin Ct., Fremont, CA 94538, 510-683-3300, Session 3B

Sean Long, National Semiconductor, 2900 Semiconductor Dr., Santa Clara, CA 95052, 408-721-2879, Session D10E

Bob Lorentzen, BottomLine Ventures, 2082 Fieldcrest Dr., Milpitas, CA 95035, 408-262-7780, Tutorial T2D

Kenneth Lowe, Sierra Semiconductor, 2075 N. Capitol Ave., San Jose, CA 95132, 408-263-9300, Session 2D

Mark Lowe, Adaptec, 691 S. Milpitas Blvd., Milpitas, CA 95035, 408-957-6664, Session 3C

Duncan MacVicar, MacVicar Associates, 1171 Buckingham Dr, Los Altos, CA 94024, 415-962-8053, Tutorial T1B

Arne Maeland, VMETRO A/S, Nedre Rommen 5, Oslo 0988, Norway, +4722 106090, Technology Fair

Tets Maniwa, Integrated System Design, 5150 El Camino Real, Suite D31, Los Altos, CA 94022, 415-903-0503, Session 4A

Kamal Mansharamani, DCM Data Systems, 4th Floor, Vikrant Tower 4, Rajendra Place, New Delhi, India 110008, +5719-96776, Session 3B

Mark McClear, Texas Instruments, PO Box 655303, M/S 8323, 8330 LBJ Freeway, Dallas, TX 75265, 214-997-5261, Session 2A

Bert McComas, In-Stat, Inc., 7418 E. Helm Dr., Scottsdale, AZ 85260, 602-483-4448, Sessions 3D, 4B

Jim Medeiros, Ziatech Corp., 1050 Southwood Dr., San Luis Obispo, CA 93401, 805-541-0488, Session D8A

Webster Meier, National Semiconductor Corp., 2900 Semiconductor Dr., MS: E-200, POB 58090, Santa Clara, CA 95052-8090, 408-721-2581, Session 4D

Eric Mentzer, Intel, 1900 Prairie City Rd., Folsom, CA 95630, Session 1C

Tim Miller, Digital Semiconductor, DEC, 77 Reed Rd., MS HL02-2/N7, Hudson, MA 01749-2895, 508-568-4122, Session 4B

John Monti, Symphony Laboratories, 4000 Burton Dr., Santa Clara, CA 95054, 408-986-1701, Session D4E

David Moore, Digital Equipment Corp., 129 Parker St., PK02/J60, Maynard, MA 01754-2571, 508-493-2257, Session D8A

Clare Moulton, Mercury Computer Systems, 199 Riverneck Dr., Chelmsford, MA 01824, 508-256-1300, Session 3A

Hans Muller, CERN Division ECP, CH-1211, Geneva 2, Switzerland, +41-22 767 3533, Session D10A

Jim Murashige, Adaptec, 691 S. Milpitas Blvd., Milpitas, CA 95035, 408-957-4813, Session 4A.1

Andy Najda, Number Nine Visual Technology, 18 Hartwell Ave., Lexington, MA 02173, 617-674-8513, Session D4B

Thomas Nygaard, VMETRO, Inc., 16010 Barker's Point Ln. #575, Houston, TX 77079, 713-584-0728, Session 5D

Richard O'Connor, Newbridge Microsystems, 603 March Rd., Kanata, ON, Canada, K2K 2M5, 613-592-0714, Session 4D

Peter Oh, Everex Systems, Inc., 5020 Brandin Ct., Fremont, CA 94538, 510-498-1111, Session 2C

Stephan Ohr, Computer Design Magazine, 316 N. Chestnut St., Westfield, NJ 07090, 908-232-1380, Session 3C

Greg Paley, Award Software, 777 E. Middlefield Rd., Mountain View, CA 94043-4023, 415-968-4433, Session 5A

Tere Parnell, LAN Times, 441 E. Bay Blvd., Suite 100, Provo, UT 84606, 801-342-6820, Session 2B

Peter Passaretti, Initio Corp., 2901 Tasman Dr., Suite 201, Santa Clara, CA 95054, 408-988-1919, Session 3B

Ed Pastor, Digital Equipment Corp., 2 Results Way, Mail Stop MR02-1/D7, Marlboro, MA 01752, 508-467-6404, Session 3A

Chau Pham, Motorola Computer Group, 2900 S. Diablo Way, Tempe, AZ 85282, 602-438-3723, Session 4B

Lisa Piper, AT&T Microelectronics, 555 Union Blvd., Mail Stop, Allentown, PA 18103, 610-712-7790, Session 4C

Dan Pirro, Teknor Microsystems, 614 Cure Boivin, Boisbriand, QU, Canada, J7G 2A7, 514-437-5682, Session D10E

Fred Pollack, Intel, 5200 NE Elam Young Pkwy, MS JF1-91, Hillsboro, OR 97124, 408-765-4423, Session D3C

Liam Quinn, Thomas-Conrad Corp., 1908-R Kramer Lane, Austin, TX 78758, 512-836-1935, Tutorial T2A, Session D9B

Richard Quinnell, EDN Magazine, 729 Clubhouse Dr., Aptos, CA 95003, 408-685-8028, Session 6A

N.Gopal Reddy, Centre for Development of Advanced Computing, Pune University Campus, Pune 411 007, India, +91(212)332531, Session 4D

Tracy Richardson, Digital Semiconductor, 77 Reed Rd., Hudson, MA 01749, 508-568-5103, Session 4D

Dave Ridgeway, Xilinx Inc., 2100 Logic Dr., San Jose, CA 95124, 408-879-4671, Sessions D3D, 4A

Jack Roberts, Dataquest Inc., 1290 Ridder Park Dr., San Jose, CA 95131, 408-437-8539, Session 1B

Jim Roche, Newbridge Microsystems, 603 March Road, Kanata, ON, Canada, K2K 2M5, 613-592-0714, Session 4D

Bernie Rosenthal, AMCC, 6195 Lusk Blvd., San Diego, CA, 92121, 619-450-9333, Session 1C

J. Scott Runner, Synopsys, 700 E. Middlefield Rd., Mountain View, CA 94043, 415-694-4214, Session D3A

Chris Russell, Cirrus Logic, 3100 W. Warren Ave., Fremont, CA 94538, 510-226-2358, Session 5B

Mike Salameh, PLX Technology, 625 Clyde Ave., Mountain View, CA 94043, 415-960-0448, Session D9A

Sam Sanyal, VLSI Technology, 1109 McKay Dr., MS 33, San Jose, CA 95131, 408-922-5371, Session 4A

Frank Schapfel, Digital Semiconductor, 77 Reed Rd., Hudson, MA 01749, 508-568-4122, Session 2D

Mark Scheitrum, CPU Technology, 47212 Mission Falls Ct., Fremont, CA 94539, 510-713-3500, Session 1A

Jim Schooler, Racal InterLan, 155 Swanson Road, Boxborough, MA 01719, 508-263-9929, Session 4C

Lewis Schrock, Compaq Computer Corp., 20555 State Highway 249, Mail Stop 120713, Houston, TX 77070, 713-378-1405, Session 2C

Tom Shanley, MindShare, Inc., 2202 Buttercup Dr., Richardson, TX 75082, 214-231-2216, Tutorial T1A, Session 5D

Dan Sheppard, AST Computer, Inc., 16215 Alton Pkwy., Irvine, CA 92718, 714-727-4141, Session 2C

Sam Sirisena, Promise Technology, 1460 Koll Circle, San Jose, CA 95112, 408-452-0948, Session 3C

Michael Slater, Micro Design Resources, 874 Gravenstein Hwy S., Suite 14, Sebastopol, CA 95472, 707-824-4004, Session 6A

Brian Small, QuickLogic, 2933 Bunker Hill Lane, Santa Clara, CA 95054, 408-987-2010, Session 4A

Ron Smith, Intel PCI Components Division, Mail Stop FM2-70, 1900 Prairie City Rd., Folsom, CA 95630, 916-356-5305, Keynote Presentation

Gene Smarte, PC Graphics & Video, 201 E. Sandpointe Av., Suite 600, Santa Ana, CA 92707, 714-513-8640, Session 2D

Jeff Solliday, Solliday Engineering Corp., 1756 18th St., San Francisco, CA 94107, 415-621-0616, Tutorial T1D

Lisa Spiegelman, Investor's Business Daily, 1270 Oakmead Pkwy., Suite 212, Sunnyvale, CA 94086, 408-720-2103, Session 2C

Margit Stearns, Symbios Logic, 1630 Aeroplaza Dr., Colorado Springs, CO 80916, 719-573-3573, Session D9C

Jeff Stockdale, Cogent Data Technologies, P.O. Box 926, Friday Harbor, WA 98250, 360-378-2929, Session 4C

Steve Tobak, OPTi, Inc., 2525 Walsh Ave., Santa Clara, CA 95051, 408-486-8243, Sessions D5D, 2A

Bao Tran, Teknor Microsystems, 616 Cure Boivin, Boisbriand, QU, Canada, J7G 2A7, 514-437-5682, Session 4B

Matt Trask, Communica, 118 Waterhouse Rd., Bourne, MA 02532, 508-759-6714, Session 1A

Neil Trevett, 3Dlabs, Inc., 2010 N. First St., Suite 403, San Jose, CA 95131, 408-436-3456, Session 2D

Lorne Trottier, Matrox, 1055 St. Regis Blvd., Dorval, QU, Canada, H9P 2T4, 514-685-2630, Session 1B

Leonard Tsai, Elitegroup Computer Systems, 45225 Northport Ct., Fremont, CA 94538, 510-226-7333, Session 4B

Tony Turgeon, Dolphin Interconnect, 3625 E. Thousand Oaks, Suite 216, Westlake Village, CA 91362, 805-371-9493, Session D10A

Jose Valdes, LSI Logic Corp., 1525 McCarthy Blvd., MS G-715, Milpitas, CA 95035, 408-433-6882, Session 4A

Arie Van Praag, CERN. div. ECP, 1211 Geneva 23, Switzerland, +41 22 767 5034, Session D7C

Rajesh Vashist, Adaptec Inc., 691 S. Milpitas Blvd., Milpitas, CA 95035, 408-957-4869, Session 6A

Deborah Vogt, Digital Semiconductor, 77 Reed Rd., Hudson, MA 01749, 508-568-4000, Session 4C

John Williams, AMCC, 6195 Lusk Blvd., San Diego, CA 92121, 619-535-4239, Session 2A

Ron Wilson, EE Times, 2800 Campus Dr., San Mateo, CA 94403, 415-525-4406, Session 1C

Lee Wilson, IBM, 11400 Burnet Rd., Mail Stop 4357, Austin, TX 78758, 512-838-6569, Session 5C

Mike Winchell, Symbios Logic, 1630 Aeroplaza Dr., Colorado Springs, CO 80916, 719-573-3573, Session D9C

Mark Wodyka, AITech International, 47971 Fremont Blvd., Fremont, CA 94538, 510-226-8960, x113, Session 5B

Martin Won, Altera Corporation, 3 W. Plumeria, San Jose, CA 95134, 408-894-7877, Session 4A

Tony Wutka, Texas Instruments, 8330 LBJ Freeway, Center 3, Mail Stop 8323, Dallas, TX 75243, 214-997-3659, Session D4A

PCI

CONFERENCE PROCEEDINGS

ISBN 0-929392-27-2

Annabooks