

# PCI SPRING


DEVELOPERS' CONFERENCE AND EXPO

APRIL 29-MAY 3, 1996

SAN JOSE, CALIFORNIA

## CONFERENCE PROCEEDINGS

SPONSORED BY

 *Annabooks*<sup>TM</sup>

PC<sup>2</sup> Consulting



# PCI SPRING

DEVELOPERS' CONFERENCE AND EXPO

APRIL 29-MAY 3, 1996

SAN JOSE, CALIFORNIA

## CONFERENCE PROCEEDINGS

SPONSORED BY



PC<sup>2</sup> Consulting

0-929392-34-5

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

ISBN 0-929392-34-5

**Proceedings of PCI SPRING '96**  
**Developers' Conference and Expo**  
April 29-May 3, 1996 • San Jose, California

Preface	ix
<i>Ed Solari, PC<sup>2</sup> Consulting LLC</i>	
<b>Session F1</b>	
Docking for Mobile Computing	1
<i>Harish Nayak, Cirrus Logic</i>	
The New Digital Media	7
<i>Tony Sheberman, Intel Corporation</i>	
How to Implement a CardBus Solution	8
<i>Gary Gildersleeve, Cirrus Logic</i>	
<b>Session F2</b>	
PCI Technology for Industrial Control Systems—Benefits and Issues	12
<i>Clyde Thomas, Allen-Bradley Company, Inc., Rockwell Automation</i>	
Using the PCI Bus for Packet Switching Applications	13
<i>Raymond Kolment, Teknor Industrial Computers</i>	
Impact of PCI Technology on Control Solutions	17
<i>Edwin Lee, Pro-Log Corporation</i>	
Leveraging PCI in Data Acquisition Applications	22
<i>Richard J. Burk, Data Translation, Inc.</i>	
<b>Session F3</b>	
Efficient Use of PCI	26
<i>Frank Hady, Intel Corporation</i>	
<b>Session F4</b>	
The Role of CardBus in a PCI Bus Hierarchy	44
<i>Claude A. Cruz, National Semiconductor Corporation</i>	
<b>Session F7</b>	
Where Do I Plug the Cable? Solving the Logical-Physical Slot Numbering Problem	51
<i>Jeff Autor and Alan Goodrum, Compaq Computer Corporation</i>	
<b>Session F8</b>	
PowerPC™ Platform	61
<i>Mike Becker, Motorola</i>	
<b>Session F9</b>	
The Standard in PCI to PCI Bridges	71
<i>Tracy Richardson, Digital Semiconductor</i>	
Design Issues for PCI-to-PCI Bridges	86
<i>Thomas L. Anderson and Mark W. Knecht, Virtual Chips, Inc.;</i> <i>Jacques Wong, Advanced Micro Devices</i>	

PCI Interrupt Controller for Industry Standard PCI-ISA Bus Architecture Using PCI-to-PCI Bridge Technology	93
<i>Ross L. Armstrong, Digital Equipment Corporation</i>	
DCM's PCI-to-PCI Bridge Solution	101
<i>Kamal Mansharamani, DCM DataSystems</i>	
<b>Session F12</b>	
PC-DMA and PCI: New Open Standard Blends Both	109
<i>Dwight D. Riley, Compaq Computer Corporation</i>	
<b>Session F13</b>	
A PCI Accelerator Architecture for the ADI SHARC DSP	119
<i>Joseph A. Sgro, Alacron, Inc.</i>	
PMC: The PCI Mezzanine Card	154
<i>Rodger H. Hosking, Pentek, Inc.</i>	
DSP and I/O System Integration for PCI	160
<i>Jack Carter and Manish Kasliwal, Sonitech International Inc.</i>	
<b>Session F14</b>	
RACEway Interlink as a PCI Switching Fabric	161
<i>Barry Isenstein and Bob Blau, Mercury Computer Systems, Inc.</i>	
PCI Bus Switching with the PSX	171
<i>Kent Dahlgren, I-Cube Incorporated</i>	
<b>Session 1A</b>	
The Future of PCI	178
<i>Edwin Lee, Ed Lee Executive Workshop</i>	
The Future of PCI	186
<i>Bert Forbes, Ziatech Corporation</i>	
<b>Session 1B</b>	
PCI and Data Acquisition	187
<i>Jim Fitzgerald, Keithley MetraByte</i>	
Design Considerations for Data Acquisition Hardware on the PCI Bus	191
<i>Richard J. Burk, Data Translation, Inc.</i>	
The Impact of PCI on the Test and Measurement Industry	195
<i>Arthur Ryan, National Instruments</i>	
<b>Session 1C</b>	
Programmable Logic Implementations of PCI	199
<i>David Ridgeway, Xilinx</i>	
PCI Is Not Just for PCs: Embedded Systems Migrate to PCI Architecture	200
<i>Mike Salameh, PLX Technology, Inc.</i>	
<b>Session 1D</b>	
PCI Performance Analysis for High-Speed Networking	201
<i>Peter N. Glaskowsky, Integrated Device Technology, Inc.</i>	
The PCI Multi-Function Device: Benefits and Design Considerations	211
<i>Margit E. Stearns, Symbios Logic, Inc.</i>	
<b>Session 2A</b>	
XVideo Family for PCI	214
<i>Bob Goodwin, Parallax Graphics</i>	

New Generation Silicon for 3D Graphics on PCI <i>Neil Trevett, 3Dlabs Inc.</i>	215
High-Speed DRAMs for PCI Systems <i>Billy Garrett, Rambus Inc.</i>	225
<b>Session 2B</b>	
Using PCI Interface in Routers <i>Aamer Mahmood, Cisco Systems</i>	233
Serial-HIPPI Network Interfaces Using the RoadRunnerPCI ASIC <i>Michael McGowen, Essential Communications</i>	234
<b>Session 2C</b>	
Computer Makers Roundtable <i>Donald F. McCook, Dolch Computer Systems</i>	243
<b>Session 2D</b>	
PCI RAID Controllers <i>K. K. Rao, Mylex Corporation</i>	255
Embedded RAID Presentation <i>Scott Jensen, Adaptec</i>	261
Fast-40 SCSI, Pushing PCI to the Limit <i>Richard Mourn, Symbios Logic Inc.</i>	262
<b>Session 3A</b>	
What's Good & What's Bad About Unified Memory Architectures (UMA) <i>Desi Rhoden, VLSI Technology, Inc.</i>	268
<b>Session 3B</b>	
Leveraging PCI Bus Bandwidth and High Performance CPUs in Designing MPEG-1 and H.261 Video CODECs <i>Frank Schapfel, Digital Equipment Corporation</i>	274
Trimedia—The Processor for PC-Consumer Multimedia <i>Selliah Rathnam and Gert Slavenburg, Philips Semiconductors</i>	275
Multimedia Bandwidth Issues Over PCI <i>Giri Venkat, Yamaha Systems Technology, Inc.</i>	283
<b>Session 3C</b>	
Board Maker's Roundtable <i>Steve Cooper, I-Bus</i>	284
<b>Session 3D</b>	
Bus-to-Bus Connections <i>Stephan Ohr, Computer Design</i>	285
PCI and Multiprocessing <i>George P. White, Corollary, Inc.</i>	286
1394 and PCI <i>Larry Blackledge, Texas Instruments</i>	302
CompactPCI™ to STD32 <i>Jim Medeiros, Ziatech Corporation</i>	312
Serial Storage Architecture: A Low-Cost, High-Speed Serial Connection for Disk Subsystems <i>Adge Hawes, IBM Havant</i>	328

<b>Session 4A</b>	
Using a Design Foundation for Flexible and Rapid PCI Interface Development	334
<i>Leo K. Wong, Altera Corporation</i>	
A New FPGA Family for PCI Interface Designs	343
<i>Brian Small, QuickLogic Corporation</i>	
PCI Implementation Kits for ORCA FPGAs: Features and Design Considerations	347
<i>James F. Hoff, Lucent Technologies</i>	
<b>Session 4B</b>	
MPEG Bridges Using T1 Lines	353
<i>Tom Thorsteinson, Linear Systems Ltd.</i>	
High Performance Vision Processing for the PCI Bus	358
<i>Fernando Serra, Imaging Technology, Inc.</i>	
The PCI Bus and Broadcast Quality Video and Audio	366
<i>Richard A. Kupnicki, Leitch Technology</i>	
Board Improves JPEG Compression Using Pre- and Post-Compression Image Scaling	381
<i>Harold Schiefer, Ernest Yeung, Steven Hanna, and Lance Greggain, Genesis Microchip Inc.</i>	
<b>Session 4D</b>	
PCI Bus Analyzer Simplifies Systems Test & Debugging	390
<i>Thomas Nygaard, VMETRO, Inc.</i>	
PCI: The Bus That Glues?	394
<i>Mark Bronson, Aeon Systems, Inc.</i>	
Latency Issues in PowerPC Reference Platform Architectures	399
<i>Don Dingee, Motorola Computer Group</i>	
PCI Passive Backplane Technologies	409
<i>Joe Pavlat, Pro-Log Corporation</i>	
PCI Shifts in the PC Landscape	440
<i>Yong Yao, MicroDesign Resources</i>	
The ATX Form-Factor	443
<i>Tim Craven, Intel</i>	
<b>Session 5A</b>	
BIOS Boot Selection	452
<i>Frances Cohen, Phoenix Technologies Ltd.</i>	
Notebook Docking: Techniques and Considerations	454
<i>Jim Kelsey, SystemSoft Corporation</i>	
<b>Session 5C</b>	
Multimedia Roundtable	462
Bridging the PCI to a Secondary Multimedia Bus: Can We Plug and Play?	463
<i>Larry Chisvin, S3 Incorporated</i>	
<b>Session 5D</b>	
CAD Tools	464
<i>Jim Lipman, EDN</i>	

Getting Quality Products to Market Faster with a Synthesizable PCI Core <i>David L. Evans, Technical Data Freeway</i>	465
The Problem of Model Availability for Simulation of Devices and Systems <i>Dave Apte, Omniview, Inc.</i>	479
Verifying PCI Bus System at Megahertz Speed <i>Sanjay Sawant, Quickturn Design Systems</i>	485
Measuring and Optimizing Performance of PCI Based Designs <i>Venkatesh Arunarthi, Sand Microelectronics, Inc.</i>	488
A VHDL Design Approach to a Master/Target PCI Interface <i>Leo K. Wong and Martin Won, Altera Corporation; Subbu Ganesan, ZeitNet, Inc.</i>	494
<b>Late Papers</b>	
The GALNET Architecture: A PCI-Based Solution for High Performance Internetworking <i>Manuel Alba, Galileo Technology</i>	500
Using FPGAs for High-Performance PCI <i>James D. Joseph, Actel Corporation</i>	520
Author Index	525
Keyword Index	527
Participant Index	529





## Preface to PCI Spring 96 Proceedings

Prior to the development of PCI engineers developing products for the non-PC (Personal Computer) industry could only select between high performance proprietary buses or standard buses like VME and Multibus I&II. PC buses like ISA and EISA were simply insufficient for the non-PC industry. Proprietary buses by definition required development of all key hardware and software components. Components for VME buses were not always compatible. Multibus I components were compatible, but became overshadowed by Multibus II which required extensive software development.

The size of the PC industry insured a diverse set of low cost components and an unparalleled selection of software. However, the lack of easy configuration and low performance of ISA bus; and the complexity, limitations, and cost of EISA bus did not provide a long term bus to replace proprietary or other standard buses.

The existence of extensive PC compatible software, appreciation for easy system configuration, and the ever increasing ASIC functionality set the stage for a new bus standard. PCI began as a bus definition to provide an easy to configure, low cost, and high performance interconnection between PC software compatible ASICs. As it was fine tuned into a PC industry standard it was expanded to include definitions for slots and add-in cards. As PCI became integral to mainstream PCs the PCI hardware costs decreased and the functional diversity of PCI ASICs and add-in cards increased. What evolved was a new bus standard that brought together performance, building block diversity, low cost, easy configuration, and compatibility with "limitless" PC compatible software.

In the mobile environment the traditional PCMCIA standard (recently renamed PC-Card 16) is essentially an extension of low performance ISA bus with configuration enhancements and power-on installation. The recent enhancement of this standard with CardBus, brings all of the advantages of PCI to PCMCIA. CardBus is a small form factor version of PCI with the power-on installation.

Most recently, the embedded systems world has also discovered the cost and software advantages, and building block diversity of PCI.

It has become impossible for proprietary buses and standard buses like VME or Multibus I&II to compete with PCI due to the size of dynamics of the PC industry. The ever growing availability of PCI cards and slots will eventually replace all of the ISA and EISA cards and slots. Similarly, the eventual availability of combination PC-Card 16 / CardBus slots in the mobile and desktop environments will facilitate the eventual extinction of PC-Card 16. Consequently, these proceedings contain information about PCI and CardBus which are the future bus standards with the unique ability to address both the PC and non-PC industries.

Ed Solari  
PC<sup>2</sup> Consulting LLC



# Docking for Mobile Computing

*Harish Nayak, Cirrus Logic, Inc.  
Systems Technology Products (STP)*

The gap in speed, capacity and functionality that has separated desktop systems from portable computers has rapidly narrowed. Today, more and more mobile computer users are relying on their portable systems to serve their needs while on the road and at the office. In this way, they avoid the problems of file transfers and version tracking that annoy their dual-computer-using colleagues.

On the road, one can get by with the small display screens, but in the office, users want larger screens, and to be able to attach networks, laser printers, scanners and other peripherals to their portable systems.

At first, they did so using ad-hoc solutions—display screen cables connected to monitor ports, network cables to network ports, and *port replicators*, where the computer's I/O ports are replicated and consolidated into a single port-replication box. Having to unplug several cables each time one left on a business trip, then reconnect them each time one returned, proved to be discouraging. It also created an opportunity to solve the problem with “docking.”

In essence, a docking set up consists of the portable system plus a docking station to which are attached whatever peripherals the user requires. Docking's primary feature is its ability to quickly connect or disconnect the portable system from the docking station and its peripherals. But docking is far more than simply a mechanism for rapidly plugging or unplugging multiple interfaces. It must also ensure that in the process users cannot inadvertently lose or damage any data files. As such, docking approaches are both related to, and limited by, the operating system's features and functions, and the I/O buses involved.

Docking is an evolving technology. With the advances of new I/O bus technologies and operating systems, docking is also advancing toward a fully automatic, any time, capability.

## Docking Stations Today

There are a range of docking solutions in place, today. They differ in terms of their physical docking attributes, and their electrical docking requirements.

### *Physical*

A *surprise-style docking/ejection mechanism* is one of the simplest but requires that the user make sure the system is “ready” for docking or undocking. There is no fail-safe mechanism, here, that permits either the operating system or basic I/O system (BIOS) to override the operation. Hence, there is a risk of losing files.

The *VCR-style and locking-style docking/ejection mechanisms* provide a fail-safe system for docking or undocking. For undocking, an eject button or icon is pressed or selected which initiates a series of interactions between the BIOS and various hardware and software components. The result of these interactions is putting the computer into a safe undocking state. Only after the undocking is approved by all involved is the portable computer actually ejected.

### *Electrical*

In addition to the various physical manifestations of docking, there are differences in docking electrical conditions.

“Cold” docking, for example, refers to a docking scheme whereby both the computer and docking station must both be powered down before docking or undocking can take place. Afterward, the computer and docking station are powered up, and the computer must go through a boot up sequence.

A so-called “warm” docking technique permits the systems to be powered up when docking, but requires that the computer be in a suspended operational state before docking or undocking. After docking or undocking, the computer must still go through a wake-up process to restore it to an operation-ready

state, or it may require a full reboot, depending upon the operating system.

In moving toward the ideal—a fully automatic, any time, docking capability—docking technology must first progress to the “hot” docking stage. Here, the computer and docking station are both powered up, and the computer is operational. The industry is on the brink of hot docking but there will be varying degrees of less-than-fully-automatic operation for a while.

## Operating Systems and Docking

There are definite relationships between operating systems and docking capabilities. For example, portable computers running DOS and Windows 3.1 are limited to cold and warm docking. These operating systems simply lack the functional support needed for hot docking.

Windows 95, however, has provided a foundation for all three types of docking, including hot docking. Its penchant for hot docking is primarily due to its dynamic loadable drivers, device enumeration, and operating system-to-BIOS links for automatically adding and removing resources.

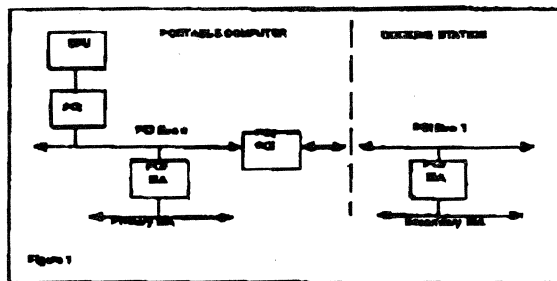
## System Buses and Docking

Designers have some choices. They can “dock” across an ISA-bus infrastructure, or do it across a PCI-bus infrastructure. There are obvious advantages to choosing PCI. It is broadband and fast with very short latency. ISA, on the other hand, is a 1980s technology, lacking in both bandwidth and speed.

However, ISA does enjoy an important advantage. It is the I/O standard for a large number of available and economical peripherals. That’s why, for now, portable computers are being built with both ISA and PCI, and it’s a good reason to equip a docking station with a secondary ISA interface, too.

When an ISA bus is present both in the computer and the PCI-based docking station, it is referred to as a “dual ISA” design. Both systems—portable computer and docking station—will take advantage of PCI-to-ISA bridging to connect ISA peripherals on both sides of the docking demarcation line via a PCI bus interface (see figure 1).

In effect, the computer’s ISA bus (primary) is connected to the docking station’s ISA bus (secondary) through the PCI bus.

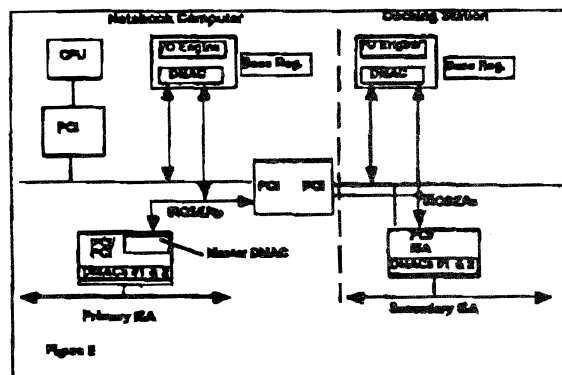


With the approach shown in figure 1, though, a DMA controller (DMAC) and Programmable Interrupt Controller (PIC) are implemented on both sides of the docking line, and they use the same I/O address space. That poses a problem.

There are two other concerns that must be addressed in implementing a dual ISA design, too. The “legacy” peripherals that use the ISA bus comply with ISA’s Interrupt Request (IRQ) specifications. And these peripherals are also designed, in most cases, to use ISA Direct Memory Access (DMA). However, neither ISA, IRQ nor ISA DMA is part of the PCI standard.

There are two open-standard mechanisms that can solve this dilemma. Serialized IRQ, or IRQSER, is a mechanism for communicating IRQ status between PCI-to-ISA bridges, and between legacy components and PCI-to-ISA bridges. Distributed DMA, or DDMA, is a mechanism for legacy DMA support on a PCI bus.

The implementation in figure 2 solves both the ISA IRQ and DMA legacy support across the PCI bus through IRQSERp and IRQSERs and the DDMA mechanism.



Here, only the PCI/ISA bridge in the computer has a master DMA controller, and all others are slaves, each having a base register and DRQ/IRQ definition.

All slave devices positively decode their DMA I/O registers. The PCI-to-PCI bridge and secondary PCI-to-ISA bridges subtractively decode the unclaimed DMA I/O registers.

Multichannel- or channel-specific “write” (e.g., not on the primary ISA bus) is broadcast by the master DMAC. Multichannel- or channel-specific “read” is also broadcast by the master DMAC. Where it is a multichannel read, the master DMAC will properly assemble the bit information, then it will return the 8-bit word during the retry cycle.

Instead of edge-triggered IRQ signals, creating a risk of glitches during docking, or a need for Q-switch isolation, the serialized IRQs are passed from PCI-to-ISA bridge via the PCI-to-PCI bridge. With this implementation, the PCICLK is stopped during docking and undocking and the IRQSER signal is ignored during those times (patent pending).

## Issues and Limitations in Docking Today

To reiterate, there have been step-wise enhancements to docking technologies as direct consequences of step-wise improvements in operating systems and I/O bus infrastructures. For example, PCI allows us a simple docking interface with high performance.

The availability and low cost of ISA-compatible peripherals, and the reality that, for now, Sound Blaster compatibility requires the ISA interface, necessitates designing the support for ISA and legacy peripherals in any serious docking solution.

Today’s operating systems, particularly Windows 95, have set the stage for hot docking by creating a foundation for it. Many of the plug-and-play supporting features of Windows 95 play significant roles in hot docking (e.g., device enumeration). However, a foundation is meant to be built upon, and the next generation of the Windows ‘95 family promises to offer an even-more-comprehensive set of features in support of hot docking among other functions.

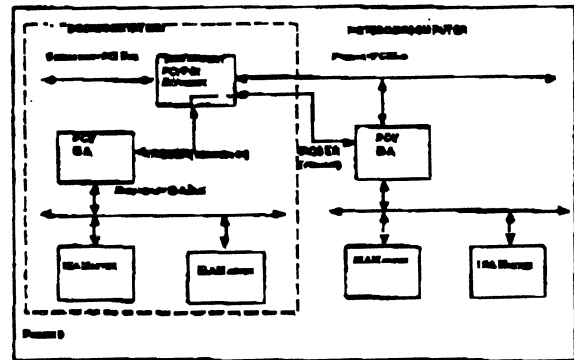
Power management is a critical feature in battery-powered portable systems. As such, power management support must be part of any full-featured docking solution.

## Solving Some Problems Addressing Design, Architecture, and Application Solutions

Given the ultimate, fully-automatic, any-time docking objective, we are now far down the road, but not quite at our destination.

As mentioned earlier, Windows ‘95 has set the stage for a step-wise leap in docking progress. It supports a dynamic loading of drivers, device enumeration, and plug-and-play. But Windows ‘95 will only enumerate the devices connected to PCI bus 0, not those connected to bus 1. For that to occur, the docking station would need a plug-and-play BIOS to enumerate its devices and interact with Windows ‘95 in order to load the drivers dynamically.

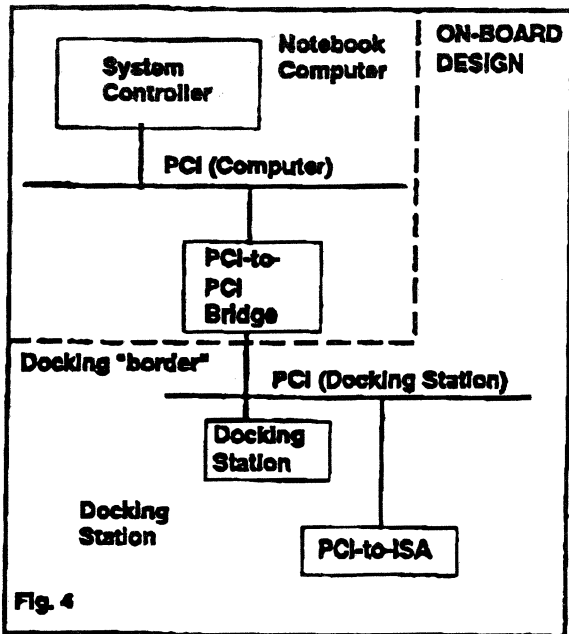
The next generation of Windows ‘95 will take care of the Bus 1 enumeration situation, but in the meantime, a transparent bus extender (patent pending) can be used to make two physical PCI buses look like a single, logical bus 0. There is no bus 0-to-bus 1 configuration cycle conversion, and no PCI configuration space involved (see figure 3).



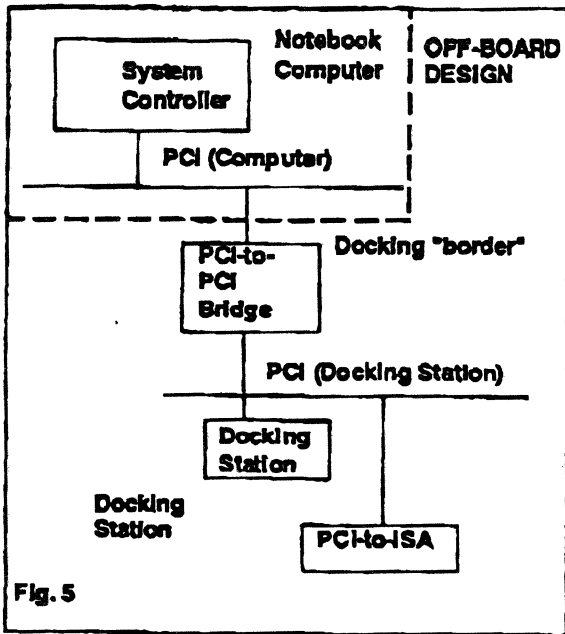
### On-Board or Off-Board?

Hot docking hardware implementations can be done in two fundamental ways—putting the PCI-to-PCI bridge on the computer, or putting it onto the docking station. There are some advantages to taking the off-board approach.

By adding the bridge to the computer, a designer has to allow more space, increase the cost, as well as the power consumption. In addition, the on-board approach may require Q-switches to isolate the IRQ signals during docking and undocking (unless a IRQSER approach is used). This, too, adds cost. For those users who do not intend to dock their systems, it is unutilized cost (see figure 4).



By putting the PCI-to-PCI bridge on the docking station (e.g., off-board), docking and undocking glitches can be isolated from on-board PCI devices without need of Q-switches (see figure 5). What's more, there is no pressure on computer board space, weight, power consumption, or cost.



### Ensuring network compliance

Since many docked computers will be part of some networking scheme, there is a need to ensure that the docked system can be compliant with network requirements.

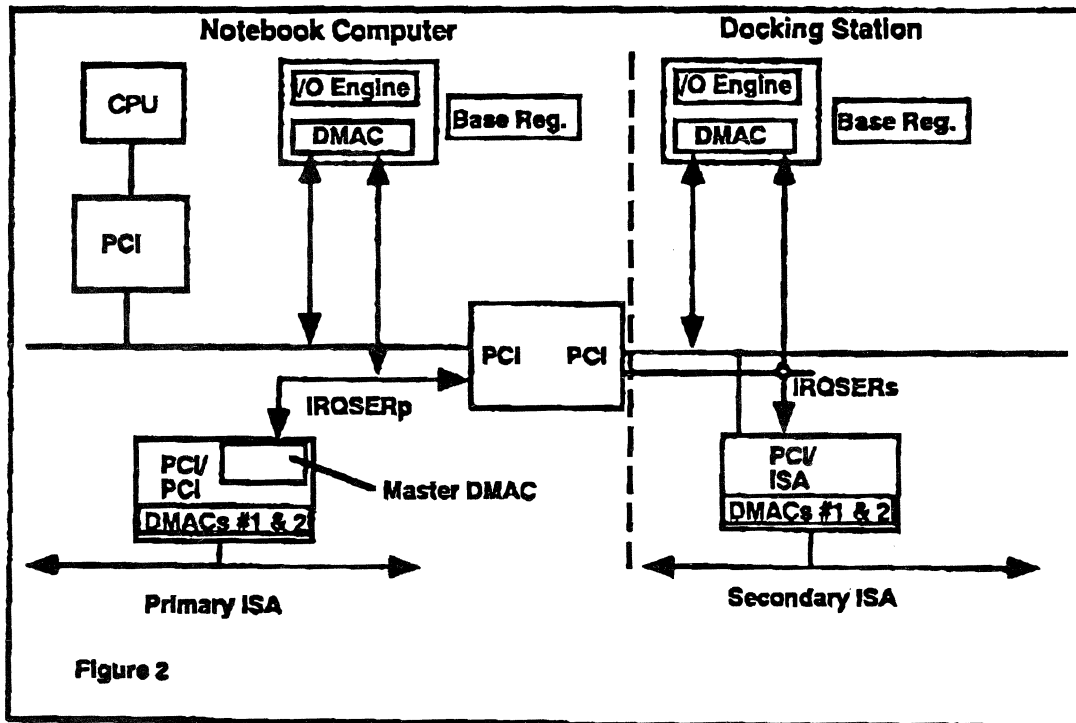
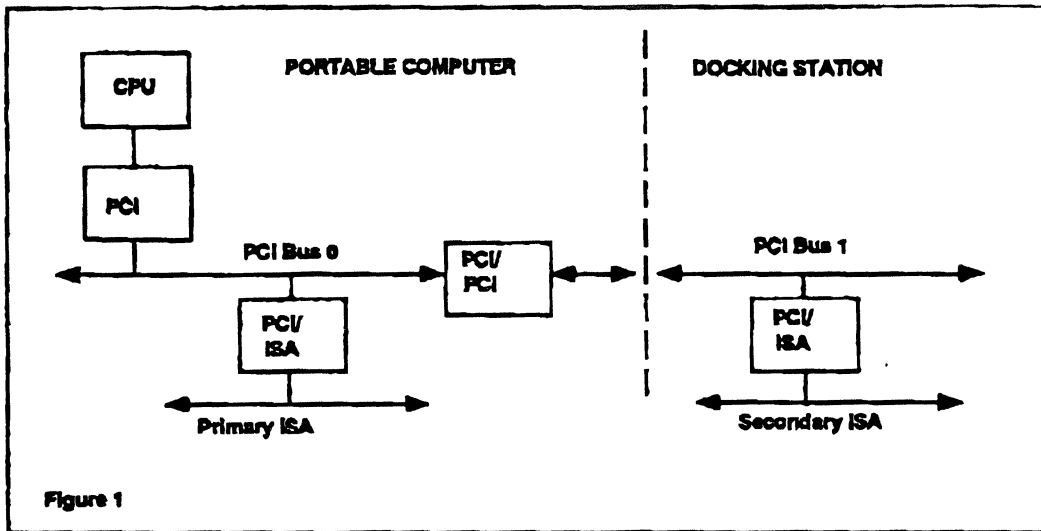
For example, a computer in sleep mode must be able to rouse to full operation within a certain time (e.g., 100 milliseconds) to meet network polling requirements. This is a significant challenge to be met by forthcoming docking implementation approaches.

### Easy docking with Windows '95

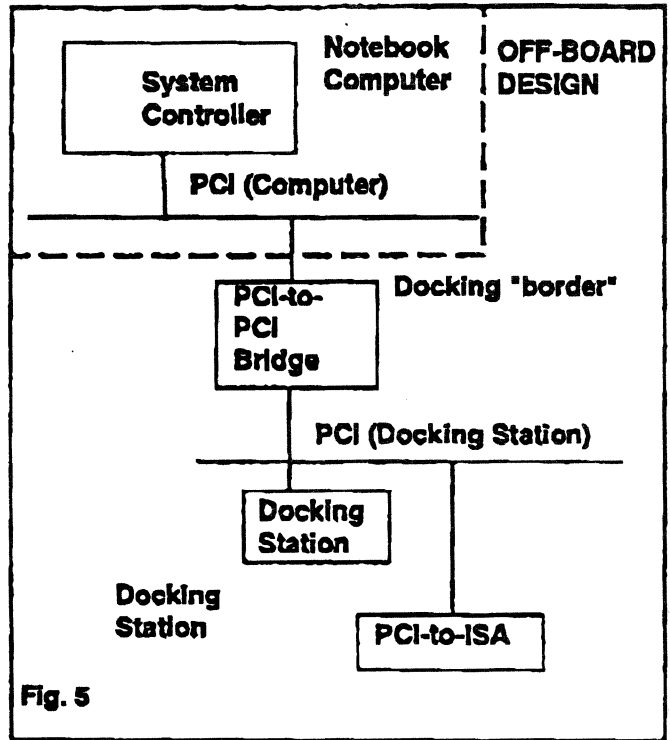
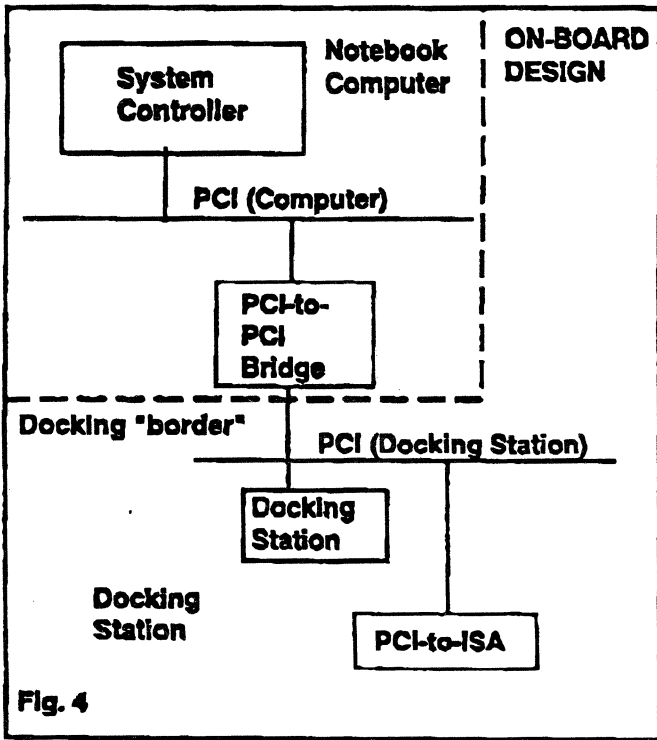
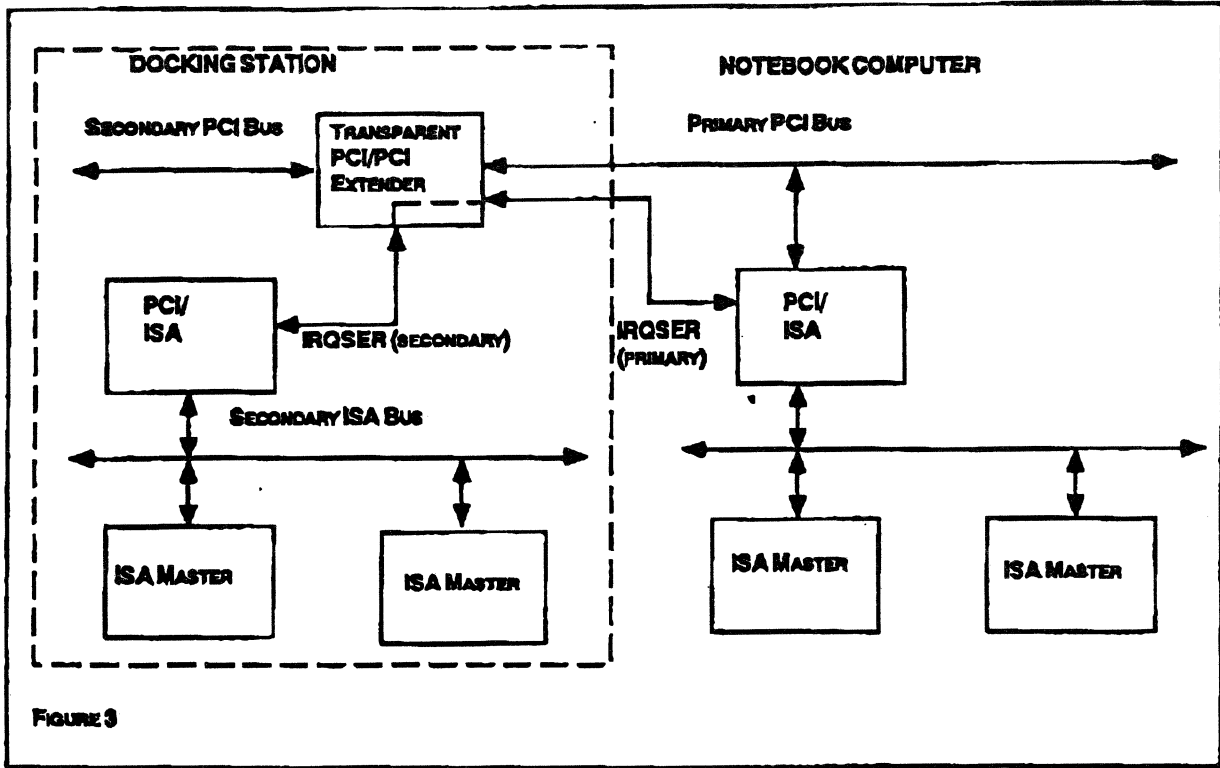
In sum, the advent of the next generation of Windows '95 and the proliferation of PCI peripherals will take us a long way toward the ideal docking infrastructure. Meanwhile, the current version of Windows '95 has already propelled us very far forward.

Despite some of its evolutionary limitations, Windows '95 is a good basis not only for hot docking, but for a hot docking scheme that is reasonably automatic.

Windows '95 is capable of supporting cold, warm and hot docking techniques, and matched by hardware that is equally capable of supporting all three modes, docking station and notebook designers will have the design flexibility and versatility they require.







## **The New Digital Media**

**Tony Sheberman  
Intel Technical Marketing Engineer  
Intel Corporation FM3-77  
1900 Prairie City Road  
Folsom, CA 95661  
(916)356-7399/2703 (fax)**

The Presentation will cover the similarities and differences of the Miniature Card to the PC Card and some typical applications for Miniature Card. The Miniature Card (Minicard) is about one fourth the size of a PCMCIA card. Typical uses include the storage and exchange of image, text, and voice data for digital cameras, audio recorders, cellular phones, handheld computers (PDAs), and other portable consumer devices. The Minicard is also the smallest standard form factor for removable memory-expansion cards. It can accommodate up to 64 MB of flash, DRAM, or ROM. The card features a 60-connection memory-only bus interface, with a 16-bit-wide, non-multiplexed data bus. Since the Minicard interface is a subset of the PC Card standard, data can be moved easily into the PC using a PC card adapter.

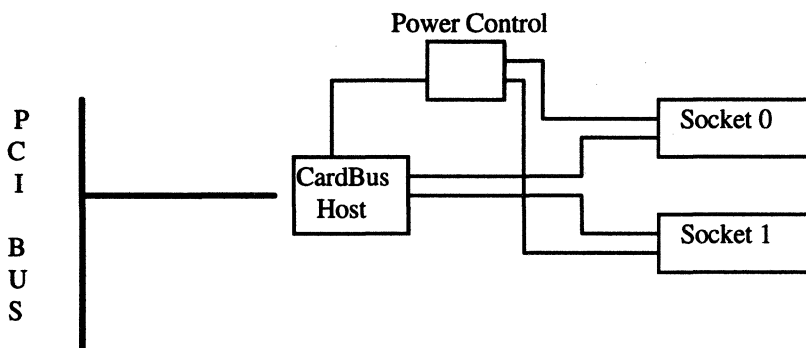
## How to implement a CardBus solution

Gary Gildersleeve  
Cirrus Logic, Inc  
3100 West Warren Ave.  
Fremont, CA 95438  
(510) 252- 6095/6080 (fax)

This article describes how to implement a CardBus bridge host controller solution and some of the design choices the architect faces. Several terms are used in this article which need a basic definition. For example, CardBus bridge controller refers to a PCI to PC Card bridge host controller. PC Card 16 refers to the revision 2.1 compatible PC Cards or R2 Cards which have an ISA type 16 bit data path. PC Card 32 refers to CardBus Cards that have a PCI type interface with a 32 bit data path.

In a basic CardBus subsystem, there are three independent interfaces, which are stated below:

- 1) Host bus bridge interface ( PCI ).
- 2) Socket interface
- 3) Socket Power control interface



For most CardBus designs, the host bridge interface signals are directly connected to the corresponding PCI bus signals. These signals are the multiplexed address/data, control and arbitration signals and interrupts. Most of the PCI bus signals are direct connections to the CardBus bridge host controller. Therefore, this article will focus on other areas of the CardBus design. There will be some discussion brought up in regard to specific signals in the component and layout section. Main areas of discussion will be the Interrupts, Power control and the optional feature of Zoomed Video (ZV). The last topic discussed is the testing and verification of the CardBus design. This article is not intended to sway the reader in anyway upon how to design their notebook system, but rather to help conger up ideas and possible problems that may occur in a CardBus design.

### *Component and layout issues*

Component placement is another design issue that should be considered in a CardBus host controller implementation. Remembering that CardBus is similar in many aspects to the PCI bus. The timing on the control signals are critical and have stringent requirements (11 ns max.). The CardBus host controller chip should be placed as closely as possible to the PC Card Connector. It is recommended that the trace length from the host bridge control to the PC Card socket does not exceed 5". Loading condition of these lines should also not exceed the loading specified in the PC Card specification. The CardBus

clock signal (CCLK/A16) should be give special attention since this is the 33 Mhz clock supplied to the PC Card socket from the CardBus host controller. To ensure a clean clock signal to the CardBus socket, a double wide trace should be used and additional guard-banding with a ground maybe preferable. To prevent Cross Talk on neighboring signals extra spacing can be added between the CCLK signal and other board signals. In the routing of the board good layout practice should be used by avoiding sharp 90 degree corners. Always use 45 degree corners instead. Since there is a lot of simultaneous switching of signals at the PCI and CardBus interfaces, adequate bypassing of the power supply is essential. This can be achieved by placing quality capacitors close to the host controller. In a PC Card 16 host controller design, placement and routing were not as critical to ensure host functionality. The PC Card 16 bus was a slow bus with few critical timing requirements. As opposed to that in CardBus host design, component placement and layout considerations are major factors that govern successful operation. . Most of the of issues stated involve basic design and board layout principles which need to be taken into account in a CardBus subsystem design.

### ***Interrupts***

CardBus host adapters typically support both ISA and PCI interrupts. ISA style interrupts are active high interrupts. These interrupts are used by PC Card 16 cards and typically are not shared between devices. Each device that requires an interrupt is assigned its own interrupt signal. PCI interrupts are active low and are designed to be shared in the system. If a CardBus card is installed in the socket which is defined as function 0, it is assigned the interrupt INTA#.

There three different mechanisms in which the ISA interrupts can be generated. These methods are listed below along with a brief description

- External Hardware to generate the individual Interrupts
- PCI/Way interrupts
- PC/PCI interrupts
- Individual Interrupt pin from the host

**External Hardware** method uses two output pins from the host (ISDAT and ISLD) and the PCI system clock. When a card interrupt is generated, the host then sends out the serial interrupt to the external hardware via the ISDAT line. Once the serial data is correctly aligned, the ISLD signal is sent to latch in the data and initiate the ISA interrupt. This is a unidirectional protocol from the host to the external hardware with no acknowledgment.

**PCI/Way** interrupt method only requires one pin (IRQSER) from the host controller and the PCI clock. This bi-directional data stream is use to communicate the state of the interrupt between the host controller and the core logic. . When an interrupt is generated by the controller, a start pulse is generated to begin the transaction. Within the start and stop time period, each interrupt is assigned three clocks which are used to show the state of the interrupt and each interrupt has its own time slot within the start and stop period.

**PC/PCI** mode supports the Mobile computing model for serial interrupts. This method requires two pins ( SOUT and SIN ) and the PCI clock to interface the SIC (serial interrupt controller). The number of interrupts supported is dependent upon the configuration of the SIC. For more information refer to the mobile computing specification.

**Individual Interrupts** means that CardBus bridge has dedicated pins for each ISA interrupt.

The choice of interrupt method used is dependent upon the host bridge and core logic that will be used in the system design.

### ***Voltage Control & Power issues***

Earlier host controller designs that only supported PC Card 16 cards could sometimes exclude mixed voltage support to the PC Card socket without being severely penalized due to the limited number of low voltage cards in the marketplace. It is no longer feasible for designers to avoid mixed voltage designs with the trend shifting towards low voltage systems and power saving. Mixed voltage support is no longer an option for the CardBus controllers. The PC Card Standard specifies that the CardBus interface can only operate at 3.3V. PC Card socket power control switches are available from many different manufactures in the marketplace and provide integrated solution for power control. These switches are used to control the Vcc and Vpp voltage levels of the PC Card socket. These switches come in either parallel or serial interface. Most of the CardBus host controllers today use the serial power control switch to free up pins.

Power requirements for the CardBus system is another area the designer needs to be aware of to determine the total system power requirements. Typically, the designer needs to know the worst case power requirements for each subsystem. For the CardBus subsystem this can be easily calculated using the following formulas.

Socket Power = (number sockets) \* ( max. voltage of the socket) \* ( Amp)

The 1 Amp value is derived from the PC Card Specification that states the maximum rating for pin of the PC Card socket is 500 ma per pin.  $2 * 500\text{mA} = 1\text{A}$

Host Power = (Highest Voltage applied to Host) \* ( 1A)

Using the formulas above, the worst case power requirements for the CardBus host subsystem would be 15 Watt of power dissipation using 5V as the maximum voltage to be used in the system.

### ***ZV (zoomed Video)***

The Zoomed Video (ZV) Port is a direct connection between a PC Card and a VGA controller / Audio DAC. It allows the PC Card to write video data directly to an input port of a graphics controller and audio data directly to a digital-to-analog converter.

A few of PC Card host adapters are being introduced in to market that are capable of supporting the proposed ZV Port standard. There are two methods of supporting ZV Port capability. The first method is termed pass through in which all the ZV Port signals pass directly through the host controller. The second method is termed "bypass" mode. Bypass mode is where the signals are re-routed from the PC Card bus directly to the video port. The video port of the graphic controller is termed the "V Port". This re-routing is accomplished by tri-stating specific PC Card Bus signals from the PC Card host adapter. Once these signals are tri-stated by the host controller during ZV Port operation, the ZV Port compliant PC Card drives video and audio data on the same signals. Video signals from the PC Card are routed to the ZV Port capable Video controller. Audio signals from the PC Card are routed to the ZV Port compliant audio DAC in the host system. This mechanism allows for an inexpensive means to add video/audio capability to a notebook or desktop system without burdening the host bus. Figure 1 shows block diagram for a typical implementation.

A ZV Port compliant PC Card, when inserted into a PC Card slot, is initialized the same way as a PC Card 16. This is specified in the PC Card standard. The ZV Port PC Card is thereafter recognized as a ZV Port card and is programmed accordingly by Card Services. In this example, the Host controller

enters into ZV Port mode by tri-stating address pins A[25..4] of the PC Card bus when the Multimedia or ZV Port enable bit is set.

The address pins are outputs from the host controller during normal PC Card operation. Tri-stating of the address pins by the adapter, allows the A[25..4] signals to carry video data and video capture timing control signals directly to a video controller and the audio signals to the audio DAC.

It should be noted that ZV Port implementations are likely to vary amongst platforms and that Socket Services software has to be customized to address these variability's. Controlling output enable inputs of the external buffers depends upon specific hardware design and Socket Services has to be aware of these specifics such as the I/O Port addresses.

**Validation and Test**

Once the design is done a very important aspect is validation and testing of the system. In most cases, the CardBus host bridge is typically the last subsystem tested and usually given the minimum time compared to other subsystems like Video controller. The CardBus interface may prove to be an even more difficult interface to validate. One reason is due to the enormous number of PC Cards in the marketplace. How can you test to ensure compatibility with every card? Also many of the PC Card 16 cards come with point enablers that bypass Socket and Card Services that can be a source of a problem. If a certain PC Card fails, how is one to determine the cause? Is the problem the CardBus bridge, the card manufacture, software, etc.. One suggestion during system validation is to start the validation of the CardBus bridge earlier. Plan on carrying out comprehensive tests to verify the bridge interface. Probe and measure the timing generated by the controller, look for timing violation, noise, Vcc and Ground bounce. Any of these problem many cause the system layout to change, and cause the design schedule to slip.

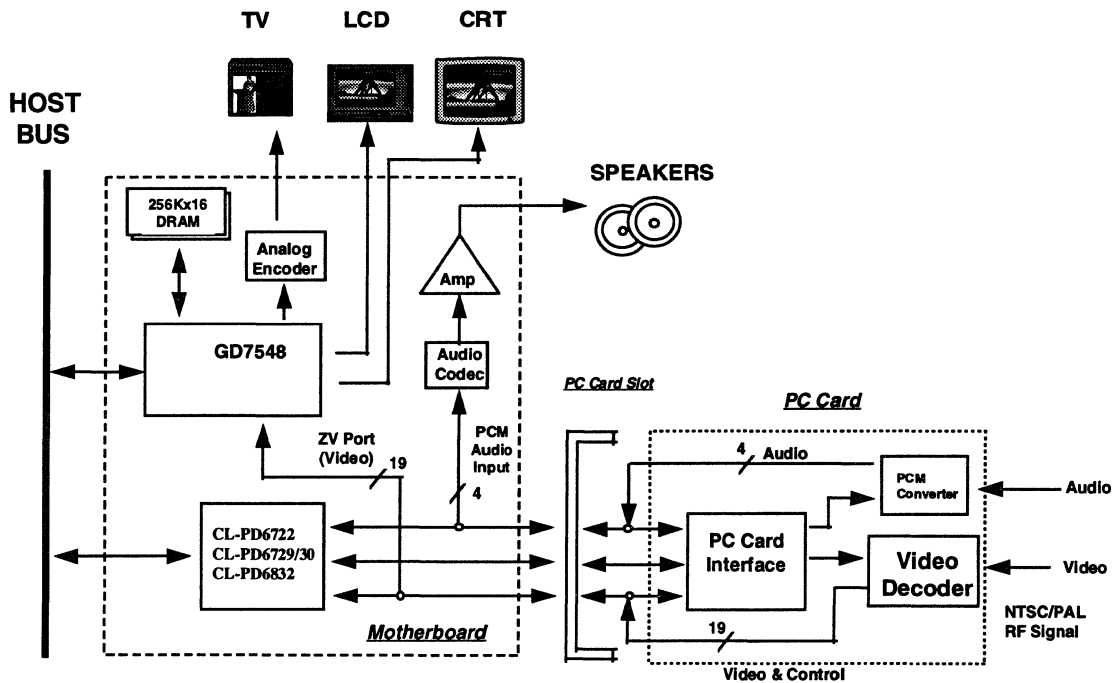


Figure 1

**Typical Example of the ZV Port Implementation**

# PCI Technology for Industrial Control Application Benefits and Issues

*Clyde Thomas*

*Allen-Bradley Company, Inc., Rockwell Automation*

Historically, the large industrial automation vendors have used proprietary bus technologies in their control solutions. A number of market and technology drivers has created interest in using standard and commercially available technologies such as PCI. This paper presents how one major control vendor, Allen-Bradley (A-B), has adopted PCI to help introduce a new line of PC-based controllers using existing A-B form factors and I/O products. The presentation will discuss the benefits of using existing PCI standards and technologies, and how the use of PCI allowed for shorter development time as well as access to additional technologies to broaden the application capability of A-B's industrial control solutions. In addition, several issues of adopting PCI technology from the commercial PC-based form factors as well as the emerging CompactPCI definitions will be addressed.

- I. Introduction to Industrial Automation Control Systems and Traditional Approaches
- II. Drivers for Change in the Industry Automation Market
- III. The Role of PCI and Its Suitability for Industrial Control
- IV. Benefits of Using PCI
- V. Unique Design Constraints
- VI. Issues Associated with Industrial Application
- VII. Close—A Trend Not a Fad

# Using the PCI Bus for Packet Switching Applications

Raymond Kolment, PCI Group Leader  
Teknor Industrial Computers Inc.

## ■ Abstract:

Present packet switching applications are normally based on the use of custom designs. The use of off the shelf PC products is generally out of the question. This paper proposes a method of designing a medium rate communication switch, using standard industrial quality products. The use and application of the PCI bus and available industrial PC products is demonstrated.

## ■ Background:

The basic architecture for most digital data communication circuits is the *T-S-T*, or Time-Space-Time data switch. This switching architecture allows messages to be handled in both the time and space domain. Most circuit switching systems and all packet switching systems use one form or another of the basic *T-S-T* architecture.

**T-Stage:** A time switch has a finite amount of memory to store incoming data packets. These data packets are subsequently routed to their intended destinations. The *T-stage* will delay the data, if necessary, to assure that there are no clashes between concurrent data packets. A packet that cannot be immediately routed will be delayed a short time before being sent to its destination. This process arbitrates the packet access to the finite output resources by scheduling the access to these resources. This process can be applied to both packet and circuit switched data systems.

**S-Stage:** A space switch provides independent concurrent cross-connections between inputs and

outputs. The classical operator switchboard, used in the earlier half of this century, is an example of an *S-stage*. Cross point switches are another example of the *S-stage*.

A *T-stage* that includes multiple inputs and outputs can perform the operation of an *S-stage*, but there are physical size limitations on this switch architecture. The number of inputs/outputs can cause an electrical implementation of the circuit to

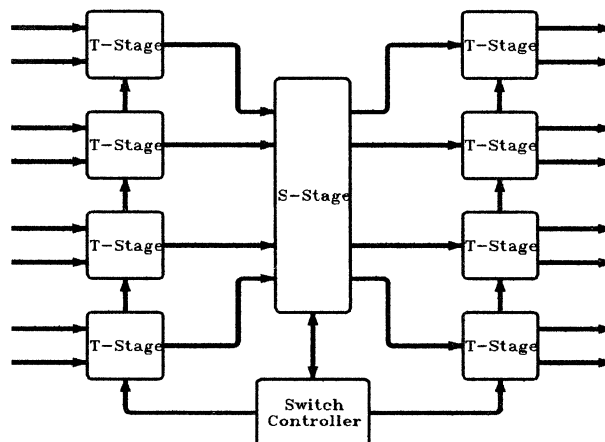


Figure 1 Typical Communication Switch Architecture.

become I/O bound, whereas the physical size of the temporary storage memory will limit the number of channels that can be handled by one circuit assembly.

This is the reason why the *T-S-T* architecture is so popular in switch designs. A simple *T-S-T* architecture allows a modular implementation for switching circuits. This modular implementation allows a single switch to be expandable in both the number of individual input/outputs that can be

serviced and the amount of data that can be handled by one central switch.

Figure 1 shows the basic architecture for a typical *T-S-T* switch. The circuit includes a *Switch Controller* that is used to monitor and control the operation of the switch. Typical functions implemented by the *Switch Controller* include:

- Switch Configuration;
- Circuit Synchronization;
- Status & Health Monitoring
- Billing & Customer Use Control;
- Circuit Switching Connection Control;
- Packet/Circuit Switch Priority Control;

## ■ Circuit Implementation:

Present implementations of this system use proprietary hardware to implement most of the switch. This can be costly to design and manufacture. In many cases, the design and manufacture of the *Switch Controller* is based on current CPU and chipset technologies. Given the constant state of flux of the CPU market, especially the chipset market, the reliable supply of CPU's as *Switch Controllers* becomes a concern.

The use of generic hardware can reduce the overall cost of such a circuit and remedy the CPU source supply problem. For instance if a standard backplane, cardcage and CPU is used, the cost of designing the *Switch Controller* is eliminated. A switch manufacturer could concentrate all of their efforts on designing switch hardware instead of spending their time redesigning *Switch Controllers*. Since the Controller is a standard product, it can be easily updated by just simply swapping boards.

## ■ PCI Bus:

By designing the *T-stage* components



# PCI Spring: Industrial Applications

of the switch as PCI compatible assemblies, the interface between *T-stages* can be greatly simplified. The actual implementation of the *S-stage* can be done with the PCI bus architecture. The PCI bus completely replaces the *S-stage*.

The PCI bus supports burst transfer rates up to 33 Mcycles/sec, with data bus widths up to 64 bits. The bus therefor yields a peak data rate of<sup>1</sup>:

$$R_{peak} = 33Mhz \times 64bits/cycle$$

$$R_{peak} = 2.112Gbits/Sec$$

Given that the bus can be used at up to 85% of its bandwidth, which is not unusual for synchronous access schemes such as *time-division-multiple-access* (TDMA<sup>ii</sup>), the net transfer rate of the PCI bus is:

$$R_{net} = 2.112Gbit/Sec \times 0.85$$

$$R_{net} = 1.795Gbit/Sec$$

As a figure of merit, one can compare this net rate to the number of telephone channels it can support. An uncompressed voice channel requires a channel rate of 64kbit/sec<sup>1</sup>. The PCI bus in this recommended application could support 33,000 simultaneous phone conversations!

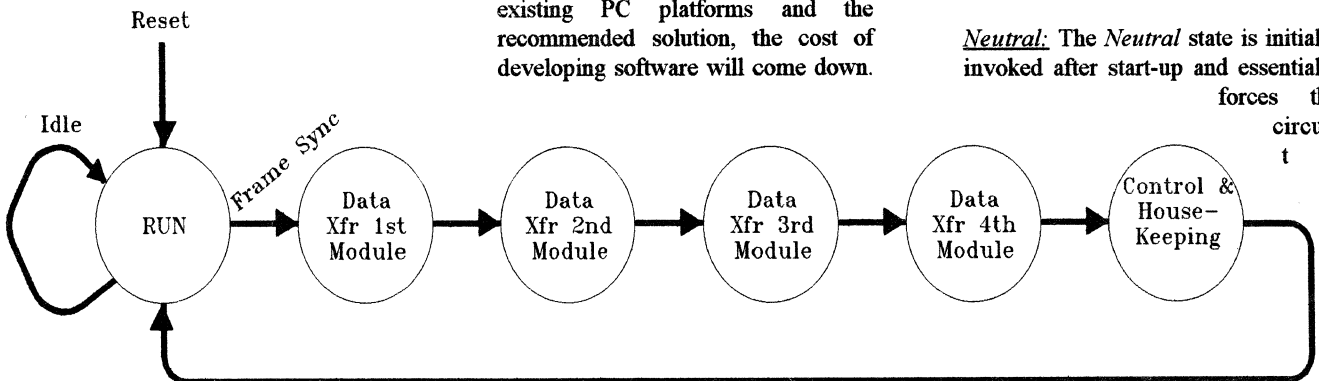


Figure 3 Transition Diagram of PCI Bus Events.

<sup>1</sup> A standard telephone service uses a sampling rate of 8kHz at 8 bits/sample.

Since the probability of using all phone lines at the same time is quite remote, a multiplication factor is used to determine the total number of lines that such a switch could handle. If the probability of a line being used is 0.2, the total number of lines that can be serviced by such a system would be over 165,000.

The use of such a circuit would find itself applicable to medium rate services. This would include such applications as PBX systems which are commonly installed in medium to large sized corporations. Since the system can support expansion by simply installing more *T-stage* elements, a common system would service many different clients.

Other medium rate services include central phone office services. As stated above, each system could handle up to 33,000 simultaneous calls.

### Software Development:

In the past, the software developed for data switching circuits was based on real time operating systems. The recommended solution discussed in this paper would maintain the use of this software database, however, the development platforms used to write the application software would be based on common PC technology. Because of the close relation between existing PC platforms and the recommended solution, the cost of developing software will come down.

This is especially evident in development and coding of common drivers used in such a system. It is also

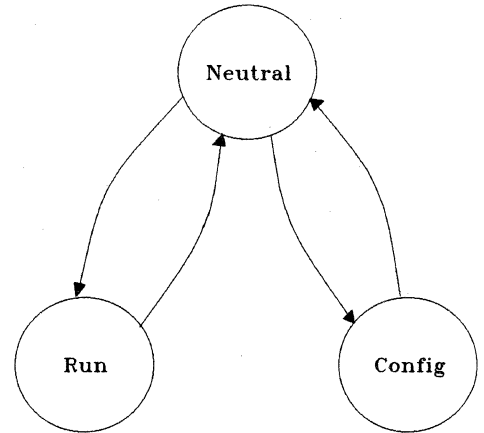


Figure 2 State Diagram of Communication Circuit.

true for hardware and software development tools.

Given the state of the art of today's PC technology and the reliable supply of industrial quality CPU's, the future development of data switches will be readily supported for years to come.

### System Architecture:

To implement the proposed system, the PCI bus must perform the same functions performed by the *S-stage* and provide interconnections between the *T-stages* & *Switch Controller*.

The circuit will assume one of three mutually exclusive states (see Figure 2).

**Neutral:** The *Neutral* state is initially invoked after start-up and essentially forces the circuit to

assume a failsafe operation mode. This mode affects all of the modules of the switch.

# PCI Spring: Industrial Applications

**Configuration:** The Configuration state is used to configure the switch. This mode is executed sequentially and is not bound by real-time operating requirements. This state may be used to perform software downloads from the Switch Controller to the T-stage modules. It may also be used to perform offline diagnostics and major switch reconfigurations.

**Run:** The Run state requires the system to operate in a synchronous mode. Figure 3 illustrates the activity during this state. The PCI must perform all of the functions of the S-stage, and must also support the communications between the system modules. This process is synchronous, and must not be interrupted by other processes within the system.

The run process is invoked from the neutral state, and is triggered by the Frame Sync interrupt. The typical period of this event is 125µS. During this period of time, the Switch Controller will command each T-stage module to send data packets to their appropriate destination T-stage. The dwell time for each module is the same. Upon completing the four transfer processes, the Switch Controller will query each T-stage for status information and send commands for the next Frame Sync cycle.

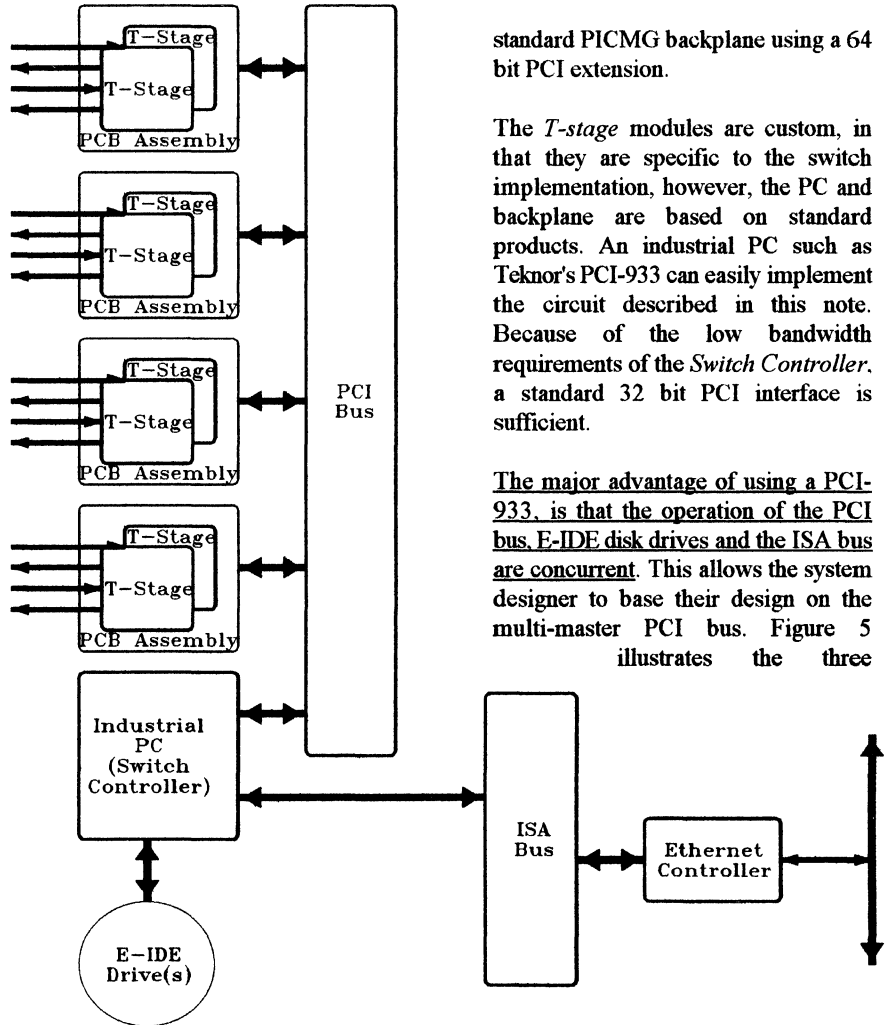


Figure 4 Physical Implementation of a T-S-T Communication Switch, Implemented with an Industrial PC and PCI bus.

standard PICMG backplane using a 64 bit PCI extension.

The T-stage modules are custom, in that they are specific to the switch implementation, however, the PC and backplane are based on standard products. An industrial PC such as Teknor's PCI-933 can easily implement the circuit described in this note. Because of the low bandwidth requirements of the Switch Controller, a standard 32 bit PCI interface is sufficient.

The major advantage of using a PCI-933, is that the operation of the PCI bus, E-IDE disk drives and the ISA bus are concurrent. This allows the system designer to base their design on the multi-master PCI bus. Figure 5 illustrates the three

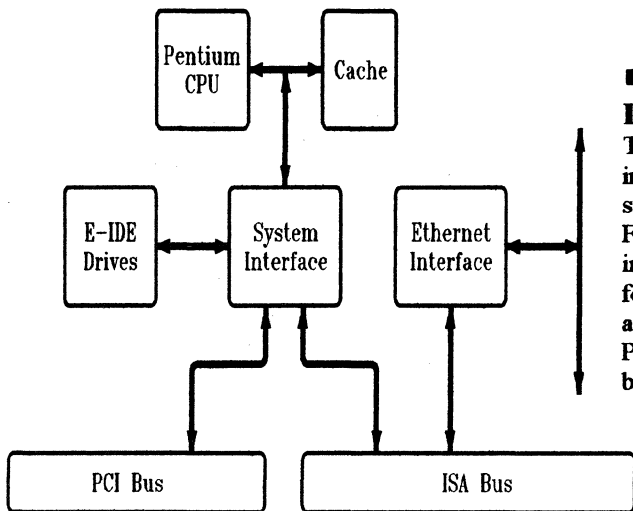


Figure 5 Block Diagram of Major Components of the Teknor PCI-933.

## Physical Implementation:

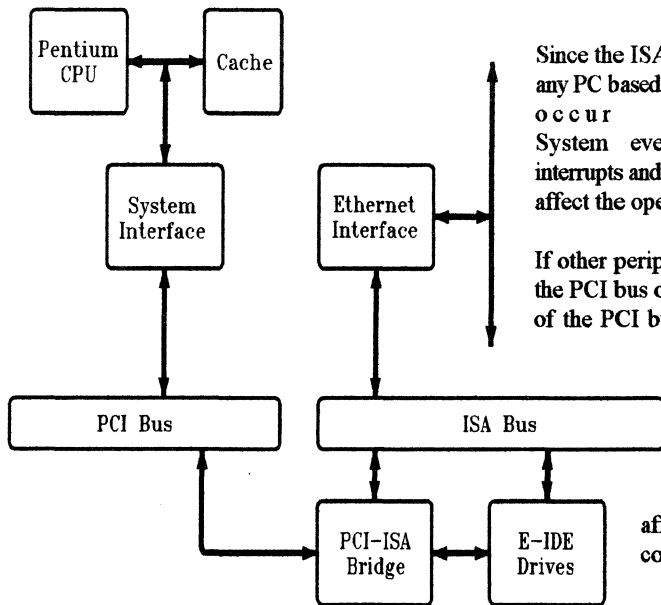
The recommended implementation of the T-S-T switch is illustrated in Figure 4. This implementation includes four PCI T-stage assemblies, an industrial PC, a PCI/ISA bus backplane (PICMG) and an Ethernet controller for system management function interface. The PCI/ISA bus backplane is implemented with a

independent paths of the PCI and ISA busses, as well as the system disks.

To maximize the switch traffic over the PCI bus, parallel processing paths must be used to assure that the PCI communication process is not interrupted.

While the system is operating in the Run state, the Switch Controller must operate independent and in parallel with the S-stage process. Asynchronous communications between the Switch Controller and the Ethernet bus must not affect the operation the PCI bus. A T-S-T communication switch based on the PCI-933 will fulfill all of these requirements.

## PCI Spring: Industrial Applications



**Figure 6** Block Diagram of Major Components of a CPU Using a PCI-ISA Bridge Interface.

Alternate products base their industrial PC designs on PCI to ISA bridge implementations (see Figure 6). These implementations have the disadvantage of locking up the PCI bus anytime the ISA bus or disk drives are accessed.

Since the ISA bus is an integral part of any PC based system, this blocking can occur quite often. System events such as real-time interrupts and refresh pulses can and do affect the operation of the PCI bus.

If other peripherals are added to either the PCI bus or ISA bus, the bandwidth of the PCI bus is directly affected. In addition to this blocking mechanism, asynchronous system events such as Ethernet and disk access will directly affect the synchronous communication process between the *T-stage* assemblies.

### ■ Conclusion:

The architecture developed in this paper demonstrates the versatility of the PCI bus architecture. Furthermore, this architecture will allow communication switch designers the flexibility and choice of using standard Industrial PC

products in their new switch designs. The overall performance of the bus is quite substantial, and is capable of taking on bigger and more complicated tasks.

The use of the Teknor PCI-933 is compatible with the needs of advanced communication circuits and is capable of handling the multi-task environment of standard switch architectures.

### ■ Biography:

Ray Kolment is the PCI Group Leader for Teknor Industrial Computers Inc. in Montreal Quebec. He holds a Masters Degree in Electrical Engineering from New Jersey Institute of Technology, and his major studies include topics in communications systems. He had completed his Master's Thesis in switching theory. Mr Kolment is presently involved in the definition and design of advanced computer and communication products at Teknor Industrial Computers Inc. 616 Curé Boivin, Boisbriand, Quebec J7G 2A7 (514)-437-5682.

### ■ References:

- i. PCI Local Bus Specification; Rev. 2.1, Oct. 21, 1994  
©PCI Special Interest Group; 1994
- ii. Local Networks; Franta, W.R & Chlamtac, Imrich  
3rd ed. D.C. Heath & Co.; Lexington Mass. 1981
- iii. A Study in Data Communication Networks; Kolment, Raymond  
Department of Electrical Engineering; New Jersey Institute of Technology  
July 1988

## Impact of PCI Technology on Control Solutions

by  
Edwin Lee (Pro-Log Corporation)

### Abstract:

PCI Technology will accelerate the decade long process of replacing systems specifically designed for industrial applications, including Allen Bradley programmable controllers and VME bus products, with systems that meet the Intel/Microsoft standards. PCI Technology will help to make the Intel/Microsoft standards as dominant in control systems as they are in desktop PCS.

PCI Technology bus speeds, I/O expand-ability, and multi-processing support are ample to concurrently handle real-time control, graphics intensive data processing, and high speed networking. The driving forces behind the move to Intel/Microsoft compatible solutions in Control Systems are: economics, the *Mind Bus*, and immediate access to the latest improvements in hardware, software, and design tools.

PCI Technology is now available for Control Solutions in three packaging formats: desktop, Passive backplane, and **CompactPCI**<sup>™1</sup>. The desktop format provides the most economic and convenient solutions at the expense of ruggedness and mean-time-to-repair. The Passive backplane format improves ruggedness and slightly reduces the mean-time-to-repair. **CompactPCI** combines the IBM PC electrical and software standards with the Eurocard packaging standards to produce cost effective systems with the ruggedness and mean-times-to-repair required by the most demanding applications.

Passive backplane PCI and **CompactPCI** are emerging, open standards supported by PICMG, the PCI Industrial Manufacturers Group. This two year old association already has over 90 member companies, and includes IBM, DEC, HP, and Force on its Board of Directors. Any company is free to make or buy products to the standards it supports.

### The Industrial Versions of PCI Technology

PCI technology is available in three packages: desktop, Passive Backplane, and CompactPCI (Eurocard).

*Desktop computers* have been used in control systems for the last decade. Although I don't have specific survey data, my estimate from experience and anecdotal data is that ~ 40% of control systems already use desktop computers because of their convenience and low costs. The trend started a decade ago. For example, in 1985, one user had already rigged a desktop IBM PC to control part of his process in a cement mixing plant. He protected the system from dust with a protective plastic covering. His backup system? His secretary's computer!

*Passive Backplane systems* have approximately the same form factor as desktop systems. However, the

---

<sup>1</sup>CompactPCI is a registered Trademark of PICMG

motherboard is replaced with a plug-in system card and a passive backplane that includes both the PCI bus and the ISA bus. In addition, the systems have beefed up cooling, beefed up power supplies and far more rugged packaging. This family of products is already available from dozens of manufacturers, including Pro-Log.

**CompactPCI** packages the desktop PC, including the PCI bus, in the Eurocard format.

**CompactPCI** has a passive backplane and a system card. However, high density pin and sleeve connectors replace the card edge connectors of the desktop packages. The cards sizes are standardized in 3U and/or 6U Eurocard formats. The cards are locked in place and are supported on all four edges. The Eurocard packaging, required in Europe for industrial systems and popularized in this country by VME bus, vastly improves shock and vibration tolerances and thermal characteristics. The pin and sleeve connectors used by **CompactPCI** enhance grounding and shielding which improve performance margins and PCI bus fanout (A system card can drive 7 peripheral cards for each set of bridge chips) and reduce EMI radiation and susceptibility.

Users of **CompactPCI** can buy or make products that bring I/O out the front panel (as is now typical for Industrial Control systems) or out through a connector to the backplane (as required by Telecommunications systems to minimize down time during card replacement).

### **The Economics of PCI Technology**

PCI Technology is driven by the >\$150 billion desktop PC market. This juggernaut is driving chip development, chip production, applications development and software development. Products used in this market have such an overwhelming volume that their costs to produce are the lowest possible. Furthermore, they are supplied by low margin, aggressively competitive suppliers.

By contrast the Controls market is somewhere around \$4 billion (Including telecommunications, industrial control, instrumentation, and medical electronics). The income stream from the Controls market is not adequate to sustain leading edge product development (hardware or software) or to produce products at competitive prices. Furthermore, the traditional suppliers require high margins to support expensive technical support, sales and service infrastructures, and to earn reasonable profits. The result is product costs to users that are two to five times that of comparable desktop products.

Apple computer, with its 7% share of the desktop PC market, has a far bigger market than the entire controls market. However, it hasn't been able to thrive by competing with the Intel/Microsoft standards. Motorola has given up on its CPU race with Intel. The income stream for the 680x0 CPUs produced by Apple, VME bus, and a captive market was not enough to sustain innovation. The Power PC is Motorola's *fig leaf*, not a viable alternative.

Just to clarify the economic perspective: \$1.5 billion is the entire market for VME bus hardware, software and systems this year (fewer than 250 thousand VME bus systems). It is also Intel's market share of the PCI system logic chip sets (40 million)! Intel is only one of several suppliers.

### **The Mind Bus and PCI Technology**

The *Mind Bus* is a term I use to describe a standard set of skills, expectations, and beliefs about computers held by the hundreds of millions of people who buy and use them. The Mind Bus has been created by the desktop PC market over the last 15 years. It is shared by engineers, executives, students and housewives (just to name a few). It's responsible for Apple's shrinking market share and with Allen

Bradley's difficulties over the last decade.

The Mind Bus provides common expectations and value references. These expectations and value references didn't exist fifteen years ago. They already impact customer preferences in Control Solutions, and explain the wide use of desktop PC's in control systems. Because the PCI Technology removes performance restrictions, the Mind Bus will dominate how designers implement Control Solutions within five years.

PCI Technology is part of the desktop PC standards and it is part and parcel of the Mind Bus. I don't have to sell it or explain it in any detail to engineers or to executives. However, I would have to spend considerable time and money to sell an alternative to Mind Bus skills, expectations, and beliefs. Just ask Apple. They are clinging to less than 8% of the market with products that may be easier to use, but don't fit mind bus standards. Within a few years we should see the same situation in Control solutions.

### **Relevant Beliefs of the Mind Bus**

*Computers are commodities, not esoteric products that require careful selection, special training, and annual service contracts.* Significant elements of this core belief include:

*I expect industry standard computers to be cheap and reliable* The best buys and latest innovations are always found in open-architecture, dominant standards supplied by many competing suppliers. Closed systems dominated computers until 1982. But, since then Wang, Apple, and IBM simply couldn't keep up with the rate of innovation and cost reductions provided by a host of suppliers vigorously competing to supply the IBM PC standards.

*I can configure my own system to meet my specific needs by using standard "plug-in" hardware and software. I expect plug and play capabilities.* Users routinely buy and successfully install third party modems, printers, and scanners. They no longer need to buy all products from a single supplier, or have suppliers install products or configure systems to specific applications. The customer thereby assigns little or no value to system configuration and system installation. Since customers can also update operating systems and applications software, they assign little value to these traditional, supplier furnished services.

*I can successfully use them without studying user manuals or paying for special training by the manufacturer.* User friendly software, built-in tutorials, third party books, or third party courses and workshops educate customers instead of User Manuals and manufacturer training.

*I can buy computers, peripherals, and software through distribution (retail) and get the lowest prices and most convenience.* Buying direct from the manufacturer is more expensive and produces less effective support.

*I can usually service my own computer with the "as needed" backup support of the manufacturer, distributor, or third party service organization when and if the need arises.* A one-year warranty supported by a telephone hotline is customary and expected. Beyond that, the failure rate is expected to be low enough that additional service is seldom needed, and annual service contracts are not cost effective.

Other core beliefs that affect the Control markets are:

*Mass produced software is relatively cheap, reliable, and user friendly. It is worthwhile to solve my*

*problem using standard software rather than paying for special purpose software.*

The desktop PC has created a value reference for software: price, performance, and user friendliness. That value reference is improving with time. Special purpose software is orders of magnitude more expensive, doesn't work as well, and is seldom as user friendly as the leading software for the PC standards. The customer asks himself: How can I use a standard word processor, accounting package, data base, customer contact package, etc. to fit my application? In the past customers would specify their needs and have software designed to meet them. That software was expensive, had bugs, and was horrible to maintain or update.

*I expect dramatic improvements in performance/dollar each and every year, therefore I want a system I can update or replace frequently.*

When a customer buys a desktop computer, she expects it to be competitively obsolete within 3 years. However, its architecture and its low costs give her the viable options to update it or replace it. The old belief was that the solution should be "competitive" for more than five years.

### **How the Mind Bus and PCI technology will alter the Controls Solutions**

*Designers will make commercial chips, operating systems, development systems, and applications software serve Control Applications.* They will accept tradeoffs from the ideal solutions because of the overwhelming economic and performance benefits of making these tradeoffs. Two examples come to mind: multi-mastering as implemented on VME bus, and *Hot Swap*.

PCI technology does not support true multi-mastering as does VME bus. On the VME bus, any CPU can take over the bus. PCI technology provides a more limited multi-mastering through a single Host that supports bus mastering for a limited number of peripheral processors. However, PCI technology has enough capability to solve any control problem. Designers will make PCI Technology fit their needs, rather than require it to add true peer-peer multi-processing.

Hot Swap, changing a plug-in card without turning power off or rebooting the system, is a Holy Grail of many control system designers. Its benefit might be to reduce mean time to repair to a matter of seconds. (I seriously doubt that anyone would actually realize this benefit.) However, unless Intel makes it a standard feature of PCI chip technology, and unless someone modifies how Plug and Play software operates, the overwhelming majority (>99%) of control systems will continue to live without it. Plug and Play software, as it works today, analyzes the peripheral cards modifies their bioses during boot-up. If a peripheral card is hot swapped there would be not assurance that it would be compatible with the system unless that system were rebooted.). Of course there is no feasible way to "hot swap" a Host CPU card.

In my opinion, we will live without Hot Swap for the foreseeable future. Let me put it another way: should VME bus, for example, successfully implement hot swap, it will not help them sustain market share in any significant way!

*Major accounts for Control Solutions will buy direct, smaller accounts will buy through Distribution.* Major suppliers will trim their overhead by focusing on shipping large quantities of fewer, standard products to key accounts. Large Distributors will be some of their key accounts. OEMs, Distributors, and third party organizations (including VARs) will provide depot level and on-site service.

### **PCI Technology and Legacy busses**

In the near-term, PCI Technology has to work with legacy buses, especially the ISA bus. These buses

have an established base of peripheral cards and operating software. PCI Technology can theoretically support as much I/O as anyone would need through PCI/PCI bridge chips. However, this solution is not yet fully implemented in the desktop world.

In the long run, PCI technology should greatly reduce, or eliminate, ISA usage in the desktop environment. It should more swiftly eliminate the use of ISA and other legacy buses, including VME, in the controls environment for a few simple reasons: reduced costs, improved performance, and greater software compatibility.

PCI Technology is supported, and will continue to be supported, by the latest in hardware and software tools. VME, for example, has different and far less up-to-date software tools to support it. It is far easier, far cheaper, and much more productive for suppliers to move their peripheral designs to the PCI bus, than to bridge the PCI bus to a legacy bus. A bridge is expensive and slows down one or both busses as it interprets one set of protocols to another. A PCI/VME bridge, for example, is like an English to Chinese interpreter passing information from one language to the other. Also, in a hybrid system of PCI and VME you can kiss *plug and play* goodbye.

What about legacy I/O busses like Allen Bradley's data highway? They'll hang on for years because old-timers will insist on sticking with what they know and will be able to hoodwink their management into paying enormous premiums to support their preferences. But new applications should move quickly to open-architecture I/O busses (like SCSI-2 or PCMCIA) supported by desktop software. There's a need for, and probably an opportunity for someone to develop an Industrial I/O bus that takes advantage of PCI technology.

## **Conclusions**

The Industrial Market is already strongly influenced by the desktop PC. Because of PCI Technology and the packaging innovations of Passive backplane PCI and **CompactPCI**, Control Solutions will increasingly depend on the products, skills, and beliefs created by the desktop PC's. In the next five years, PCI technology will become the overwhelmingly dominant computer technology in Control Solutions.



## LEVERAGING PCI IN DATA ACQUISITION APPLICATIONS

Richard J. Burk  
Data Translation, Inc.  
100 Locke Drive  
Marlboro, MA 01752  
(508) 481-3700/3080 (fax)  
e-mail: rburk@datx.com

### *ABSTRACT*

PCI's numerous performance and functional advantages are a critical benefit to data acquisition. Especially in data acquisition (DAQ) applications where users cannot compromise their data integrity, nor can they afford to compromise acquisition speed, PCI has emerged as the clear choice. Inherent design features of the PCI bus that boost performance and productivity in data acquisition include much faster bus speed, ease of installation and configuration, greater expandability and guarantee of future support. TCC Industries, a manufacturer of cellular phone accessories, recently migrated all of the company's testing PCs to PCI systems to achieve a higher degree of accuracy. TCC reduced testing time to 5 seconds using a PCI data acquisition system, compared with 12-16 seconds using a non-PCI setup. A professor from the University of Waterloo has invented a new scanning beam confocal microscope that utilizes a PCI-based DAQ board from Data Translation. This DAQ implementation would not have been possible without PCI's unique performance advantages.

### *PCI VS. ISA IN DATA ACQUISITION*

When evaluating a PC-based data acquisition system, the current state of technology leaves users faced with a choice between the ISA (industry standard architecture) bus or the newer PCI (peripheral component interconnect) bus. The numerous technical and performance advantages of the PCI bus make a PCI-based data acquisition system an easy choice, although certain applications may be more well-suited to a dedicated ISA-based system.

#### **ISA Drawbacks**

Across the ISA bus, applications can move a maximum of 400kS/sec (thousand samples per second). That is to say, no more than 400,000 data samples can be transferred across the bus -- either to or from memory -- each second. When ISA peripherals begin to push the bandwidth limits of the ISA bus, the user begins to either pay for on-board memory, or for time (seen as system delays). Data that cannot be sent immediately across the bus as soon as it comes in must be stored locally or stalled -- or it is simply lost.

A further limiting factor of this architecture is that ISA peripherals must pass all data through the CPU to system memory, consuming valuable system overhead as data travels to and from memory. While DMA (direct memory access) has been utilized to provide direct access to system memory, CPU clock cycles are still being applied to data movement, essentially stealing time from other applications and system calls. A further resource drag is the ISA memory controller itself, which grabs CPU time every time it needs to write or read from memory.

For some data acquisition applications, ISA's 400kS/sec bandwidth clearance can easily be enough, but since data acquisition applications often require bi-directional data flow, that bandwidth is quickly consumed and application performance suffers. For example, an application acquiring data from a laser microscope at 300kS/sec, and applying real-time control to an x/y-table that moves the item beneath the microscope at 200kS/sec, will quickly consume all available bandwidth and generate an unstable control loop. Dropped data bits force the application to sample data at a lower rate than the data is coming in, resulting in unstable or inaccurate readings.

#### **PCI: Ideal for DAQ**

PCI's numerous performance and functional advantages are a critical benefit to data acquisition. As data acquisition has traditionally pushed the limits of system performance, dedicated systems for data acquisition have become a pervasive mindset in the industry. The emergence of PCI-based systems is changing that mindset, and promises to open up the number of PC-based data acquisition applications. Especially in data acquisition (DAQ) applications where users cannot compromise their data integrity, nor can they afford to compromise acquisition speed, PCI has emerged as the clear choice. Inherent design features of the PCI bus that boost performance and productivity in data acquisition include much faster bus speed, ease of installation and configuration, greater expandability and guarantee of future support.

## **Speed, Cost and Time Benefits**

PCI peripherals, running asynchronously, can send data along the 32-bit bus at a rate of 66MS/sec (megasamples per second). In addition, because the PCI architecture enables peripheral boards on the bus to access systems memory directly without using the CPU, DAQ boards can be acquiring data without wasting CPU overhead. Furthermore, PCI DAQ users can be acquiring data to memory while at the same time doing analysis in real-time on existing data, all while communicating with other functions on the network.

Using PCI's bridging capabilities, multiple PCI buses can be connected, ad infinitum, with standard, off-the-shelf PCI expansion hardware. This is done via a PCI-to PCI bridge chip, which offers the additional benefit of being able to get around capacitive load limitations and expand the number of plug-in slots. This enables DAQ users to set up multiple DAQ boards, and run them all simultaneously, without hitting the PCI bandwidth ceiling. In order to achieve this kind of expansion with ISA, users would have to add additional machines to their production setup.

Furthermore, a DAQ board plugged into a PCI slot carries its own configuration information in software -- users do not need to set any jumpers or identify any base addresses -- a common headache with ISA DAQ boards. Not only does this provide extreme ease of installation and use, but because all hardware settings can be controlled in software, users can easily customize the configuration of their DAQ system at any stage of their operation.

## ***REAL-WORLD APPLICATIONS OF PCI IN DAQ***

A growing number of users are moving their data acquisition applications to the PCI bus, primarily to realize the benefit of higher throughput. Since performance in many data acquisition applications is directly dependent on throughput, or how many points of data can be dumped into system memory for analysis, the bandwidth capabilities of the bus correlate directly to testing accuracy. Marlboro, Mass.-based Data Translation's new PCI-based data acquisition product line, the PCI-EZ Series, by design, supports device input up to 1000kS/sec (or 1 megasample per second), giving an immediate 2.5 times performance increase over their ISA counterparts.

## **PCI in Telecommunications**

Joey Nieves, production engineer for TCC Industries, Inc., a Cerritos, Calif.-based manufacturer of cellular phone accessories, recently migrated all of the company's testing PCs to PCI systems to achieve a higher degree of accuracy. TCC Industries has been using PCI-based data acquisition cards from Data Translation to test and grade high-sensitivity microphones using RMS (root mean squared) analysis of voltage output, where extremely high-speed acquisition is critical.

"We were initially running two test systems and found that our PCI system was registering more accurate results. The non-PCI system was dropping data points because of bus bandwidth limitations. It quickly became imperative to upgrade all our test systems to PCI," said Nieves.

Nieves explains that during production a variety of variables are introduced which can compromise the sensitivity of the microphones, and inaccurate grading of these components will significantly impair overall product performance in the field. "A single millisecond separation in signal pick-up can give us an inaccurate reading. Our PCI-based data acquisition machines now ensure that data is transferred to memory as fast as it comes in," said Nieves. Nieves reports that testing time has been reduced to 5 seconds per unit for each microphone test, compared with 12-16 seconds each with his non-PCI setup.

Nieves is in the process of developing another application for his PCI DAQ implementation which will run a series of four tests in sequence on each finished unit. Whereas microphone testing was performed prior to final assembly in the past, Nieves now plans to skip this step until the unit is fully assembled, and test the microphone as part of the final test suite. Testing of the fully assembled units will save time and ensure a higher degree of quality control. Nieves again compared test times between two test systems and found that the four-test routine took 10-15 seconds on the PCI system, compared with 25-30 on his ISA setup. Since he must run up to 1500 units through final testing per day, the time savings is significant.

Nieves reported that it took 2 programmers less than 6 hours to develop this test suite from start to finish using Data Translation's visual programming language, DT VEE.

## PCI in Microscopy

Another DAQ user leveraging the benefits of PCI is A. E. (Ted) Dixon, Ph.D, from the University of Waterloo. Dixon has recently formed a new company, called Biomedical Photometrics, that will bring to market a new scanning beam confocal microscope that utilizes a PCI-based DAQ board from Data Translation.

Confocal microscopy is the process of shining a focused laser beam onto a specimen or subject and measuring the level of reflected light using an avalanche photo diode (APD), a highly sensitive single-point light sensor. The APD, reading grayscale only, converts light levels into analog signal levels, which are fed into Data Translation's 12-bit A/D (analog-to-digital) converter.

Data from a confocal microscopy device is "point-source" data, whereby each frame is scanned in one pixel at a time and fed into system memory. Enabled by the high bandwidth of the PCI bus, this device creates a functional imaging application from a point-source detector, producing a field size and resolution that even a high-end imaging board is not capable of.

Biomedical Photometrics' device, for which the company has coined the term MACROscope™, combines the rapid scan of a scanning beam laser microscope with the large specimen capability of a scanning stage microscope. The MACROscope proves a significant step forward in microscopy because it can scan a 25-micron-sized object, with a 0.25 micron resolution, as well as being able to scan a 7.5 cm object with 5 micron resolution, producing a zoom ratio of more than 3000.

"With this device, the PCI bus is actually enabling the advancement of microscopy," said Ted Dixon, president, Biomedical Photometrics Inc., "as this level of resolution and scanning field in the past simply required too much time to acquire data."

The new device will be particularly useful in applications where large specimens must be examined at high resolutions, and where it is necessary to examine small areas of interest in the specimen at extremely high resolutions. Possible applications include:

- biomedical, such as fluorescent gels used in gene sequencing;
- materials science, such as imaging paper fibers and coatings;
- semiconductor quality control, such as photoluminescence imaging of compound semiconductor epitaxial layers, wafers and devices; and
- forensic science, such as latent fingerprint detection, or imaging of fluorescent gels for DNA fingerprinting.

The MACROscope sends data across the PCI bus into system memory at rate of 300 kS/sec. While this data rate would not, in and of itself, push ISA bandwidth limits (400 kS/sec), the PCI bus provides the extra bandwidth necessary to run simultaneous control and analysis functions critical to the MACROscope application.

Considered a "slow-scan" system, the single frame rate for an image with a resolution of 512x512 pixels is 5 seconds, and for 2048x2048 resolution, the frame rate is 25 seconds. The higher the resolution, and the wider the scanning area, the slower the data acquisition, as both resolution and scanning area quickly increases the volume of data points being sent into the DAQ board. Based on the 2.5x improvement that the PCI bus offers, an ISA-based system would output images at rates of only 12.5 and 62.5 seconds, depending on the resolution.

Biomedical Photometrics is currently running Data Translation's 12-bit A/D converter at its top speed, 300 KHz, but for extremely high resolutions across wide scanning areas, the company will be looking to run Data Translation's next-generation converters at upwards of 3 MHz. The PCI bus ensures the viability of this growth path.

Biomedical Photometrics' implementation benefits from several PCI bus strengths, not the least of which is reduced cost because the DAQ board requires no on-board memory. "Back when we first started specifying this kind of system, DAQ boards had to have a memory buffer on the board because the ISA bus couldn't handle the data fast enough to put through to system memory, which really drove up the total cost of our system" said Dixon. "The PCI implementation gave us a way to eliminate the redundant memory and still get the throughput we needed."

PCI performance also enables the use of a 12-bit DAQ (as opposed to 8-bit), giving Biomedical Photometrics tremendous dynamic range in the data input signal. Using an 8-bit A/D converter, this type of application would typically result in data bits that are outside the useable range (either too bright or too low), effectively narrowing the dynamic range of the image data (for example, resulting in only 6 bits of real data). Using a 12-bit A/D converter ensures that even with unusable data points, the application still ends up with a 10-bit real, dynamic range (even an 8-bit dynamic range would be acceptable).

This type of dynamic range is especially critical in optical tomography, where the device takes in a series of images at different focus positions in order to compose a three-dimensional image. In the past, optical tomography required taking a series of slices, then resetting the analog gain based on the maximum and minimum values, and then going back and taking all the slices again. Data Translation's PCI-based A/D converter, an off-the-shelf 12-bit solution, eliminates this time-wasting step.

Biomedical Photometrics also went with Data Translation's DAQ board for the ability to select different frame sizes in software. "Data Translation's programmable gain feature enables our users to select different frame sizes in software depending on the particular specimen being viewed," said Dixon. "Other DAQ board implementations require setting of jumpers in hardware to change gain levels, and we wanted make this instrument as easy to use as possible." Furthermore, without programmable gain, the MACROscope would require additional optics to achieve such a high zoom ratio.

### **Conclusion**

PCI is a future trend that has gathered significant momentum in recent years, and shows no signs of letting up, while the era of ISA is quickly coming to a close. Most new PCI systems are still manufactured with ISA add-in slots, but future systems will have fewer and fewer of these slots, until they ultimately cease to exist. PCI paves the way for lower cost products, as manufacturers no longer need to include large amounts of expensive on-board memory to handle large data transfers.

The PCI bus offers many performance enhancements that make it ideal for high-bandwidth applications, and is sure to be a significant step forward in PC-based data acquisition. While not many users are buying new ISA-based PCs these days, a large number of "hand-me-down" ISA systems are making their way down the corporate ranks and into the production or testing department (typically the lowest level on the corporate PC food chain). Production managers must weigh the benefits of a new PCI system against an aging ISA system in perfect working order. A growing number of these managers are realizing that the performance benefits realized in PCI-based data acquisition applications are worth the investment. With plenty of room for growth into the foreseeable future of the PC, the PCI bus gives users the safest and most robust platform to build DAQ applications.

---

# Efficient Use of PCI

Frank Hady

Platform Architecture Labs  
Intel Corporation



## Agenda

---

- u Define PCI Efficiency
- u Charting PCI Efficiency
- u Rules for an efficient design
- u Why you should follow the rules
- u Effect of PCI to PCI Bridges
- u Conclusions

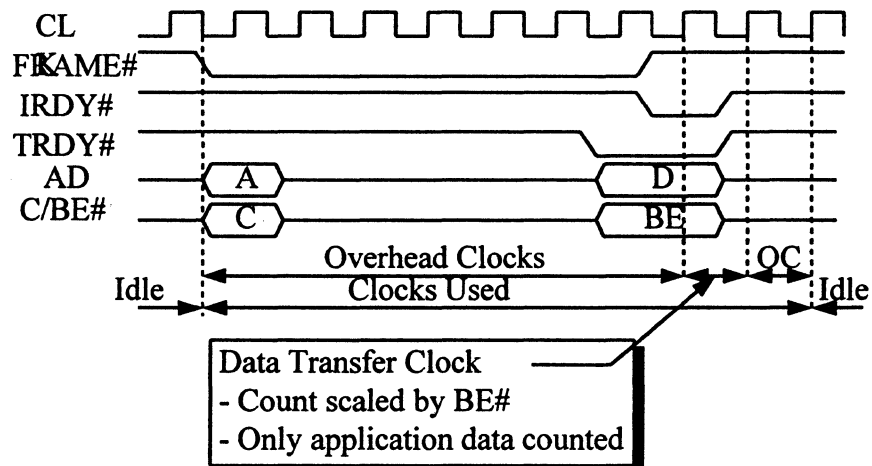


# PCI Metrics

- u Bus Utilization
  - Utilization = (Clocks Used) / (Total Clocks)
    - Clock is used if #Frame + #Irdy + #Trdy is True
- u Data Throughput
  - $\text{Thrpt}_{\text{Data}} = \text{Thrpt}_{\text{PCI}} - \text{Thrpt}_{\text{Control}}$
- u PCI Efficiency
  - Maximize  $\text{Thrpt}_{\text{Data}}$
  - Minimize utilization
  - Optimize system performance



## What Isn't Overhead?



# PCI Efficiency

---

Objective: Quantify the efficiency of moving application data over the PCI bus.

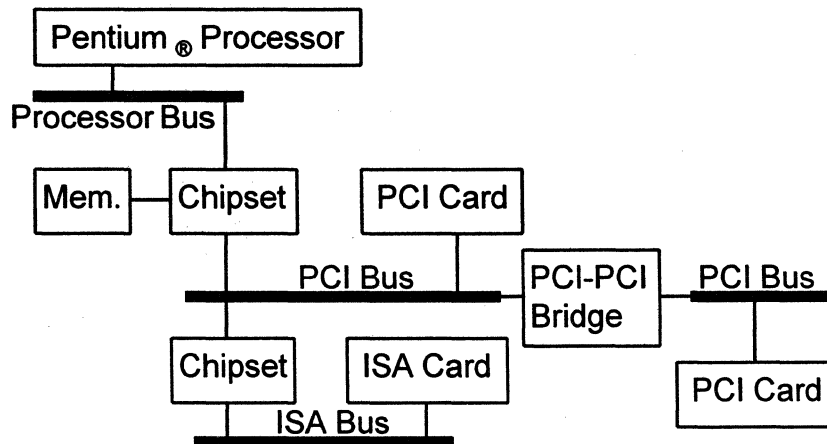
$$\text{PCI Efficiency} = \frac{\text{Data Transfer Clocks}}{\text{Clocks Used}}$$

$$\text{PCI Efficiency} = \frac{\text{Thrpt}_{\text{Data}} / (\text{PCIBusWidth})}{\text{Clocks Used}}$$

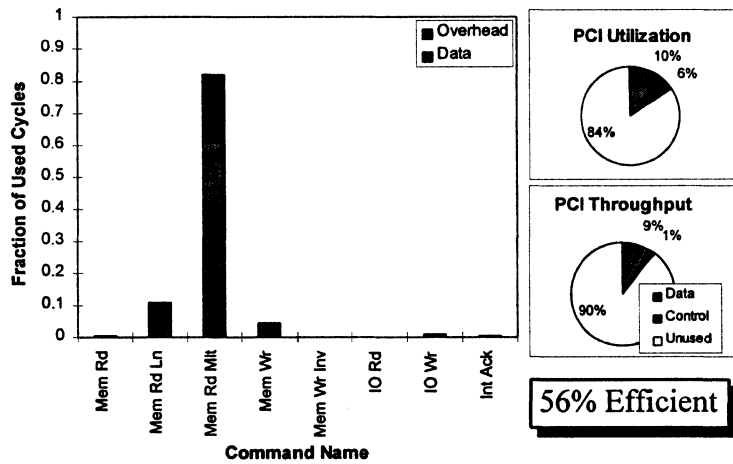


# PC Architecture

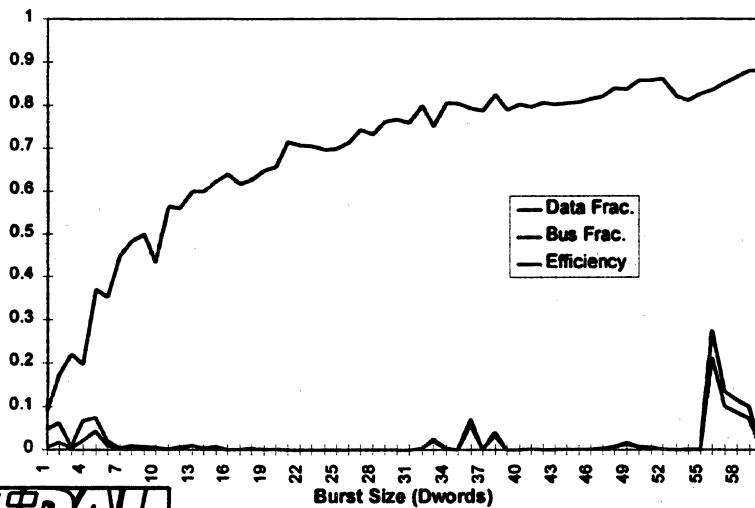
---



# PCI Efficiency Charted

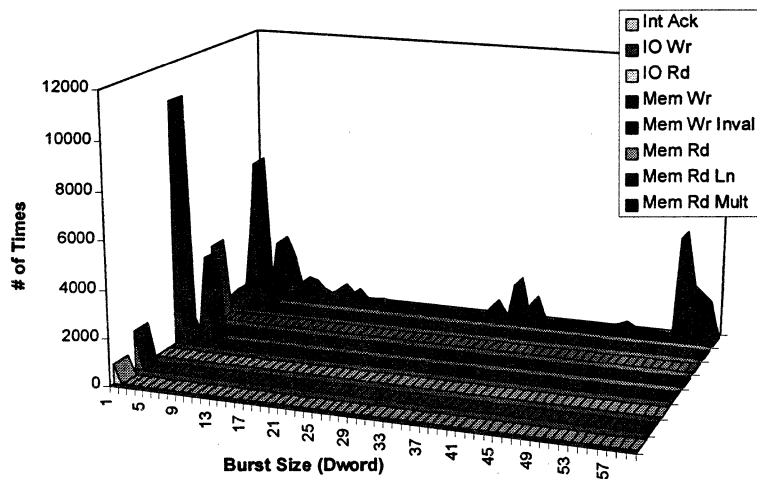


# PCI Burst Length and Efficiency





# PCI Command Usage Charted

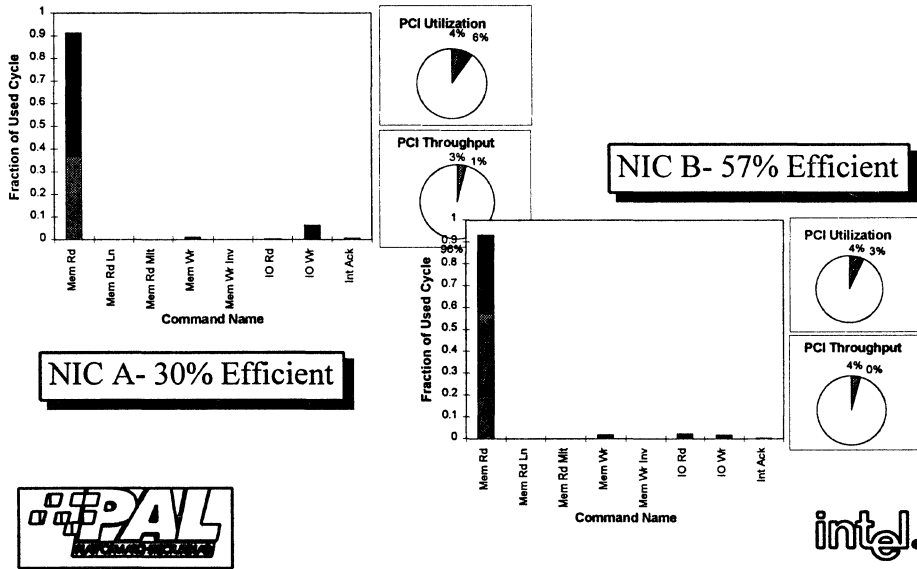


## Rules for Efficient PCI Use

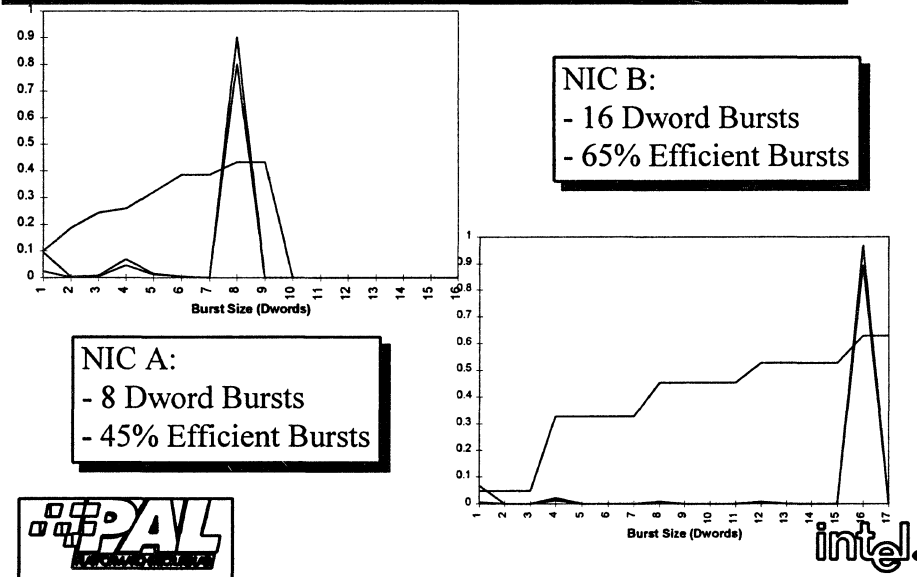
- u Use long bursts
- u Use memory commands, not I/O commands
- u Implement advanced commands
  - Mem Read Line (MRL): 1 cache line reads
  - Mem Read Mult (MRM): Multiple cache line reads
  - Mem Write Inval. (MWI): Multiple cache line writes (must be aligned)
- u Minimize latency
- u Follow the rules, not experiments



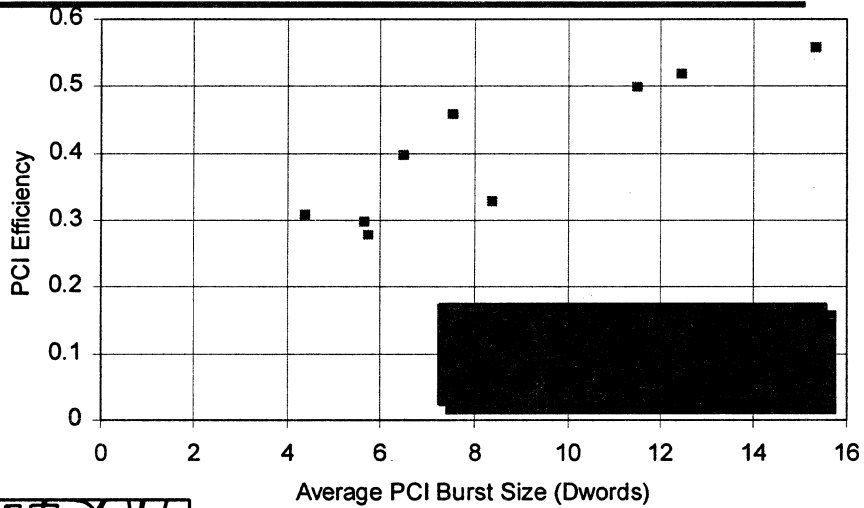
# Use Long Bursts



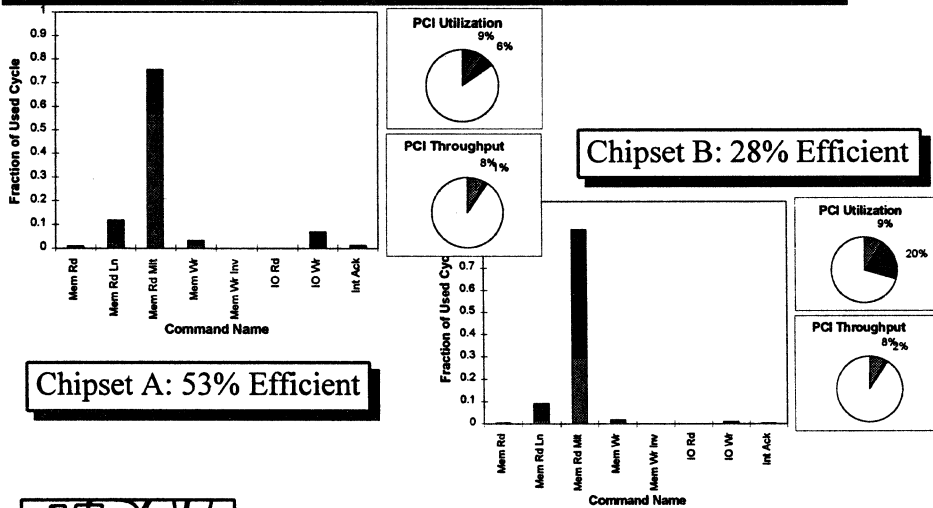
# Use Long Bursts (cont.)



# Short Bursts = Low Efficiency

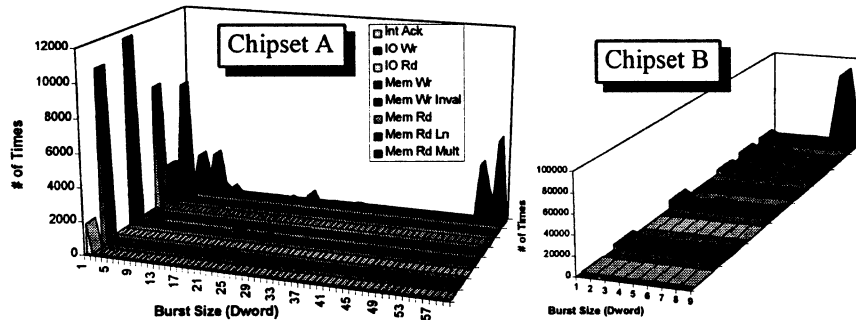


# Chipsets Can Limit Burst Length



## Chipset Limited Burst Length

---



- u Don't design to a specific chipset
- u Follow the rules to achieve high PCI efficiency on all chipsets



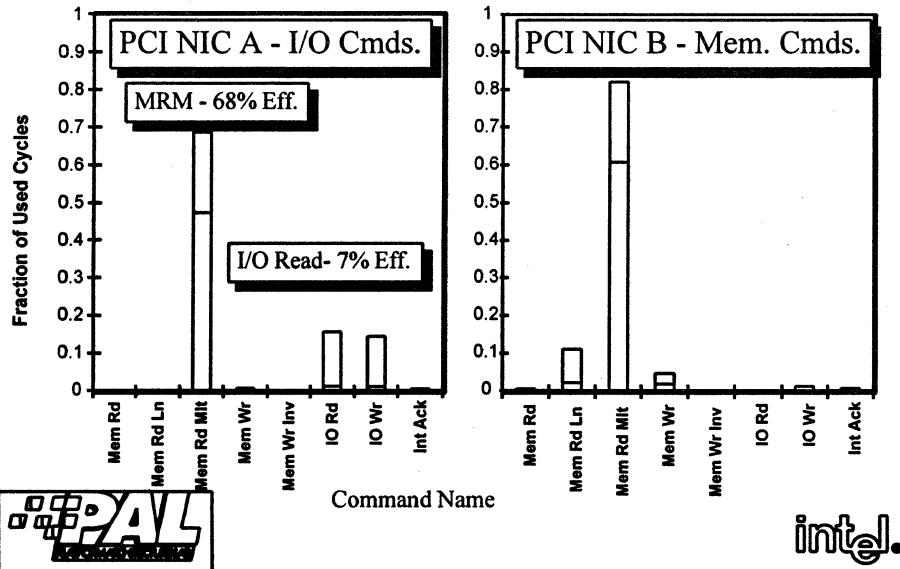
## I/O Address Space Accesses

---

- u Poor PCI Efficiency
- u Forces Ordering in System
  - May serialize CPU
  - Serializes Chipset



## Avoid Expensive I/O Commands

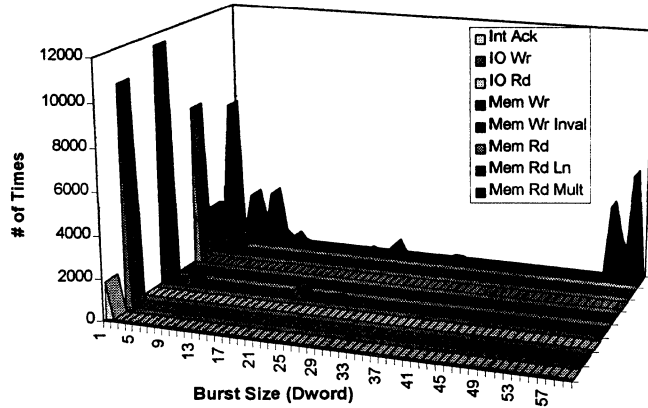


## Use Advanced Read Commands

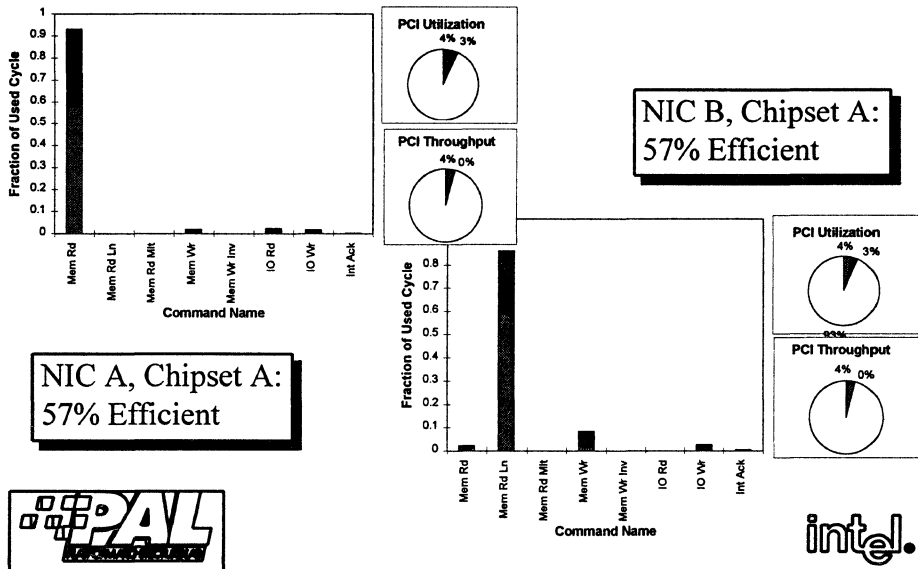
- u Memory Read (MR)
  - Short reads: 1 or 2 Dwords
  - Reads with side effects
- u Memory Read Line (MRL)
  - Medium Reads: ~ 1 cache line (8 Dwords)
- u Memory Read Multiple (MRM)
  - Long Reads: > 1 cache line



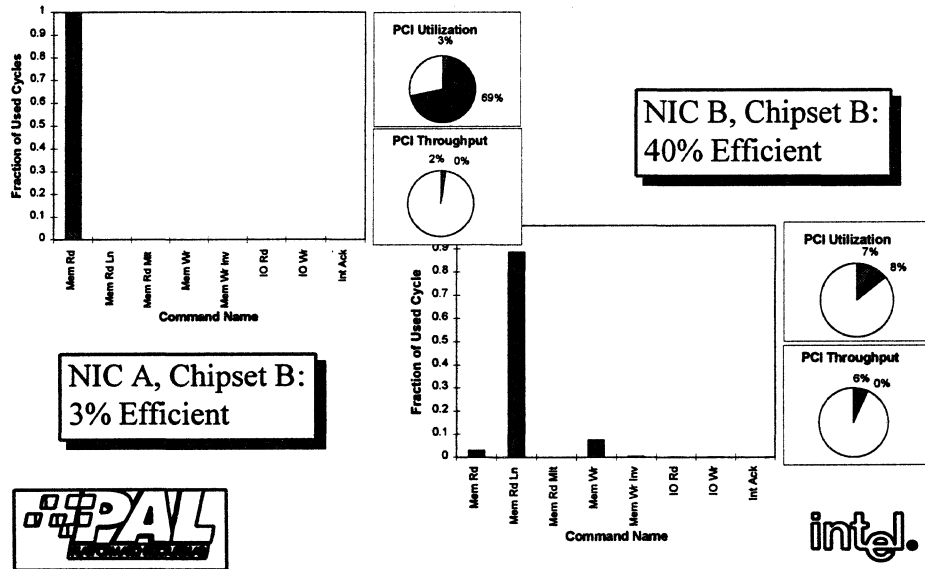
# Which Read Cmds to Use



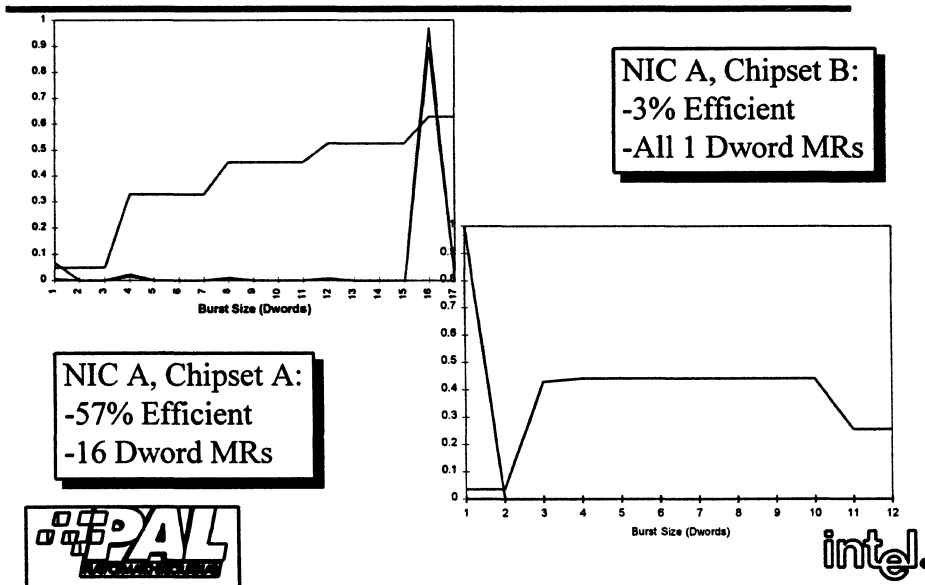
# Read Command - Chipset A



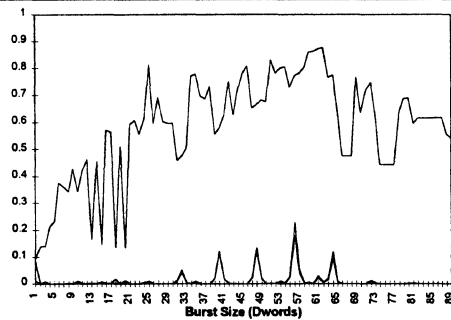
# Read Commands - Bridge B



# Read Commands - NIC A



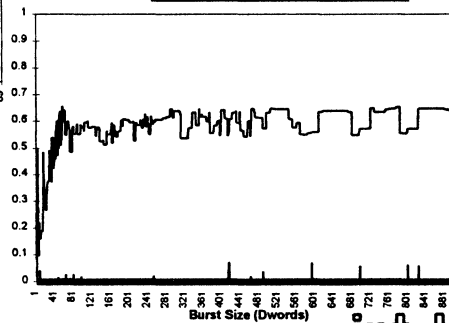
## Read Commands - NIC B



NIC B, Chipset A:  
-57% Efficient  
-Large MRL



NIC B, Chipset B:  
-40% Efficient  
-Huge MRL



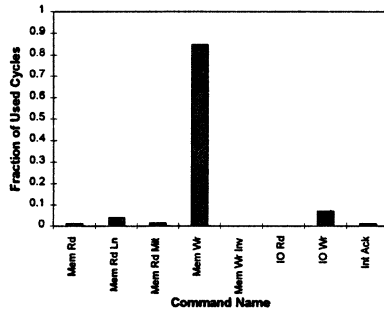
## Use Advanced Write Commands

- u Memory Write
  - Short writes: < 1 cache line
  - Long unaligned writes
  - Writes ending in incomplete cache lines
  - Do not terminate to start a MWI
- u Memory Write Invalidate
  - Long cache line aligned writes

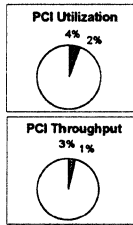




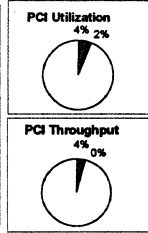
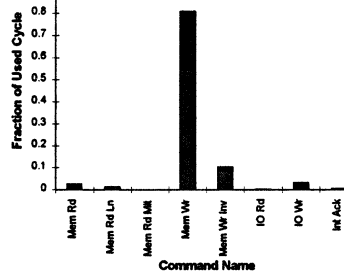
# Write Commands - Chipset A



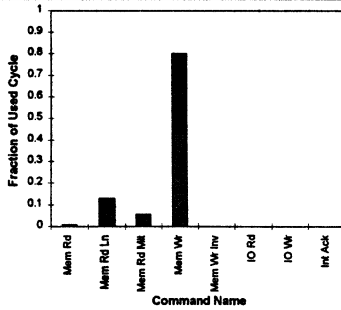
**NIC A, Chipset A:  
50% Efficient**



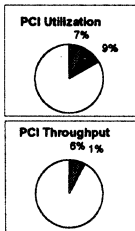
**NIC B, Chipset A:  
66% Efficient**



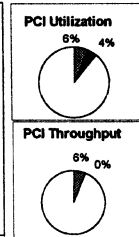
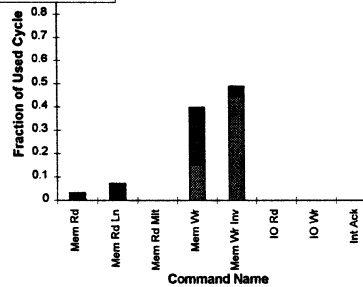
# Write Commands - Chipset B



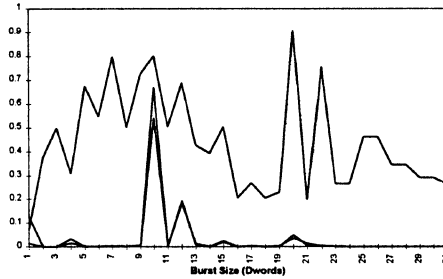
**NIC A, Chipset B:  
38% Efficient**



**NIC B, Chipset B:  
60% Efficient**



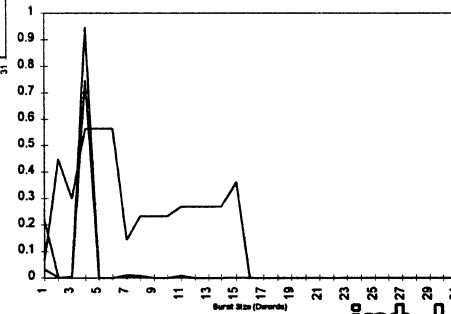
## Write Commands - NIC A



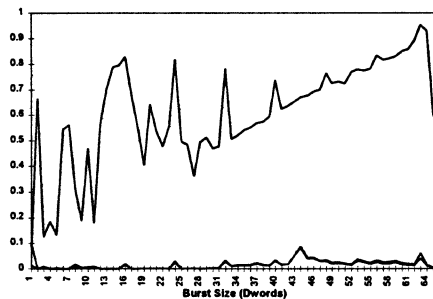
**NIC A, Chipset A:**  
 -50% Efficient  
 -10 Dword Bursts



**NIC A, Chipset B:**  
 - 38% Efficient  
 - 4 Dword Bursts



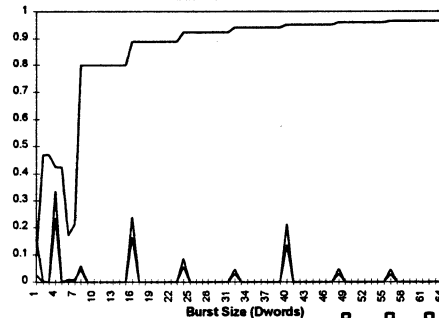
## Write Commands - NIC B



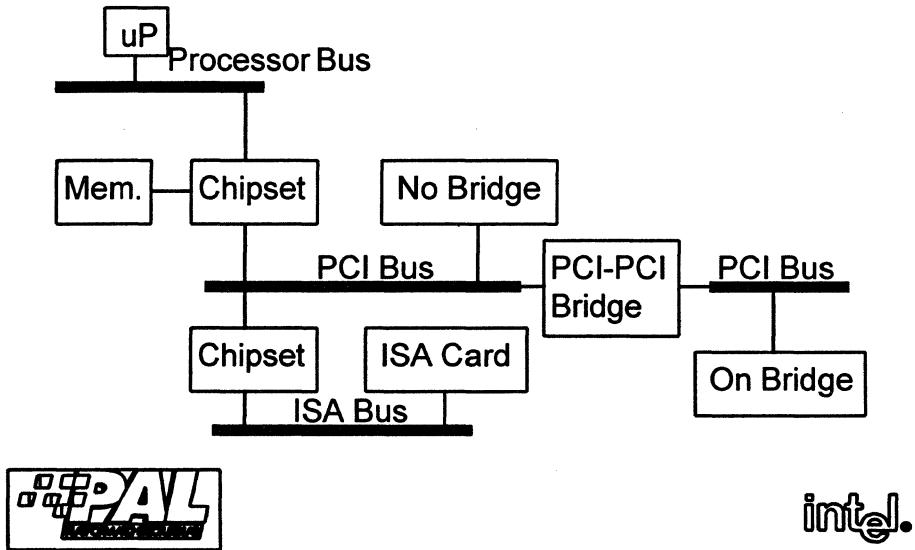
**NIC B, Chipset A:**  
 -66% Efficient  
 -Large MW and  
 MWI Bursts



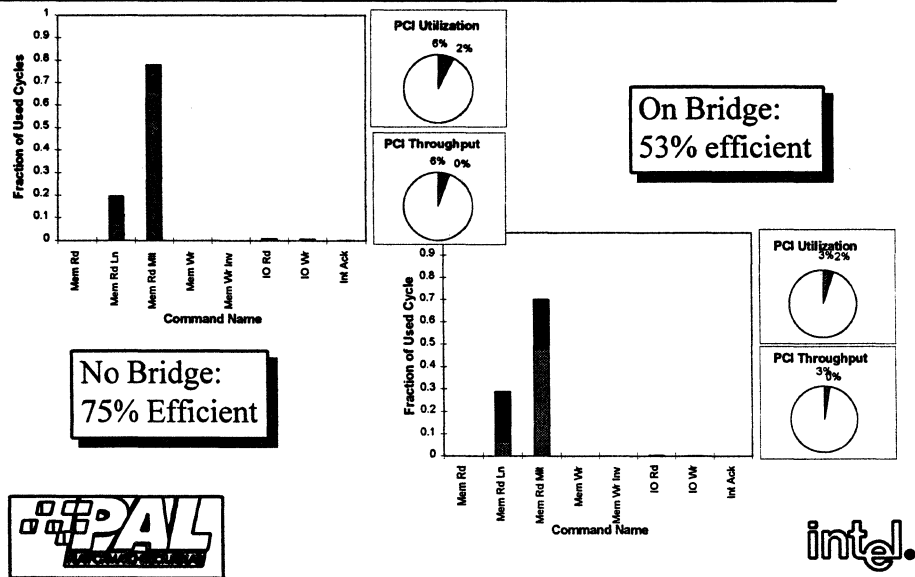
**NIC B, Chipset B:**  
 -60% Efficient  
 -Large MWI Bursts



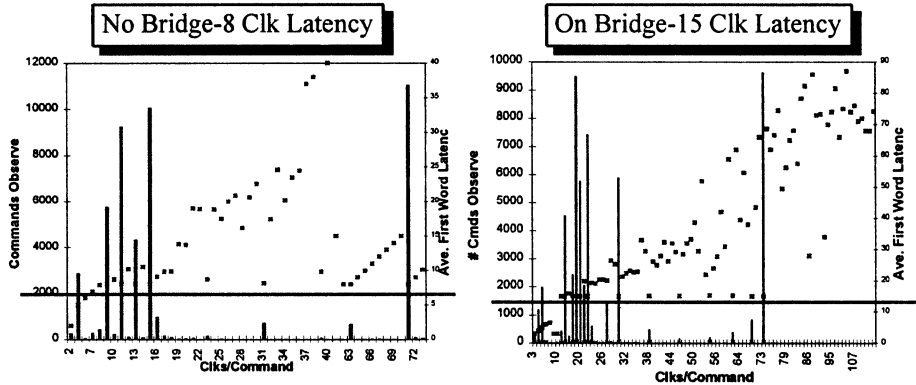
# PCI to PCI Bridges



# PCI to PCI Bridge - Reads



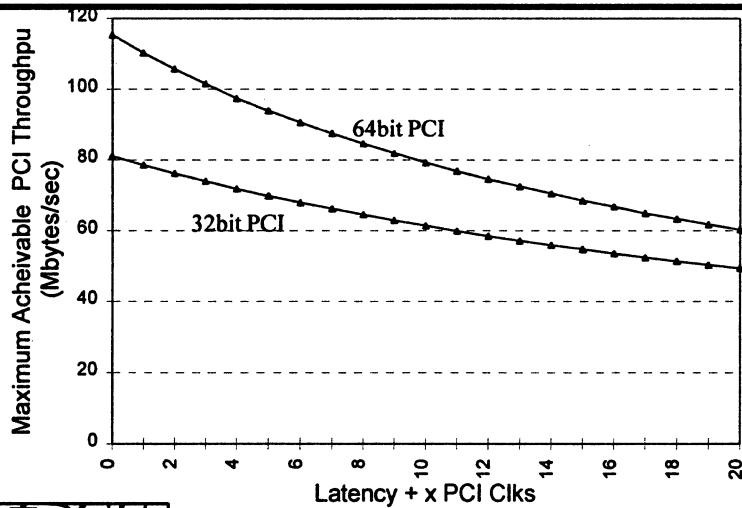
# PCI to PCI Bridge - Reads



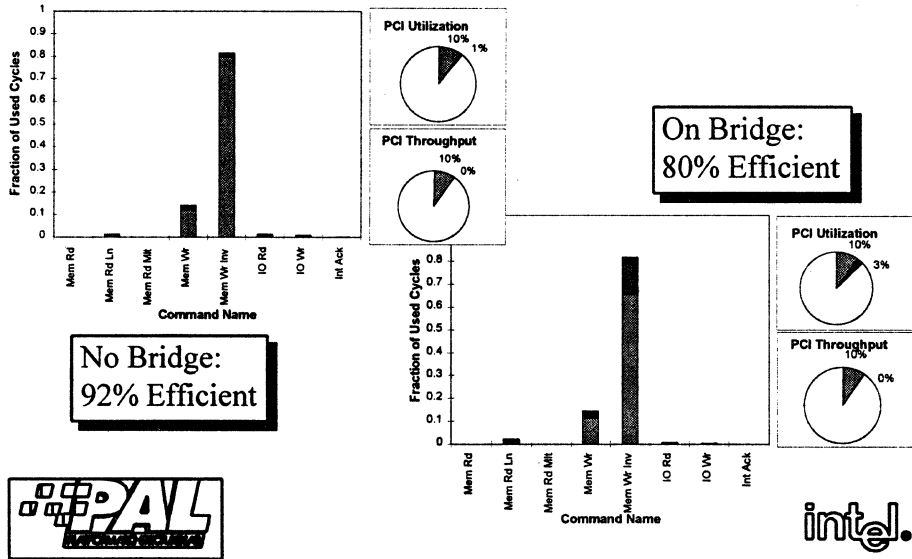
- u Adds 7 Clks in read latency dropping PCI Eff from 75% to 53%
- u Future bridges may minimize this latency, but not eliminate it



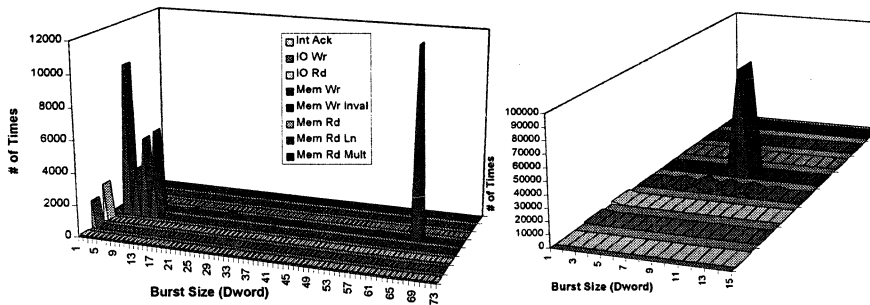
# Read Latency and 64 bit PCI



# PCI to PCI Bridge - Writes



# PCI to PCI Bridge - Writes



- u Bridge turns 65 Dword Bursts into 8 Dword bursts, dropping efficiency from 92% to 80%
- u Write posting minimizes drop in efficiency
- u Future PCI to PCI bridges may eliminate this problem



# Conclusion

---

- u PCI Efficiency is the metric (Dwords/PCIClks)
  - Optimize your designs to this metric
  - Consider other designs by this metric
- u The Chipset/PCI card combination determines PCI efficiency (Don't Optimize for a single chipset!!)
- u Current PCI-PCI bridges impact PCI Efficiency
- u Achieve high efficiency across platforms by:
  - Implementing PCI Advanced Commands
  - Use Memory Commands, not I/O Commands
  - Use long bursts
  - Minimize read start latency



# THE ROLE OF CARDBUS IN A PCI BUS HIERARCHY

Claude A. Cruz

National Semiconductor Corporation  
333 Western Avenue, M/S 10-26  
S. Portland, ME 04106  
(207) 775-8318; FAX: (207) 761-6137  
ccruz@fmis02.nsc.com

## Abstract

CardBus is a high-speed 32-bit interface defined by the PC Card standard. This point-to-point architectural “cousin” of PCI shares PCI’s signals, synchronous protocol, and performance levels. These similarities give CardBus a natural place in a PCI system’s bus hierarchy (see Figure 1).

Early CardBus implementations utilize a PCI-to-CardBus controller which is located on a platform’s level-0 PCI bus. This controller acts as a “bridge” which maps CardBus resources onto portions of a PCI-based host system’s memory, I/O and Configuration address spaces. The bridge allows PCI bus cycles to be sent “downstream” from CPU to PCI agents, or “upstream” to the CPU. Thus, the PCI-to-CardBus bridge performs the same function as a PCI-to-PCI bridge; both support the hierarchical connection of multiple PCI-protocol busses.

This paper will sketch the close relationship between CardBus and PCI, as motivation for why these two busses fill complementary roles in a PCI bus hierarchy. We will then explore three of the several possible roles of CardBus within such a bus hierarchy:

- CardBus as a link between a host-system PCI bus and a higher-level PCI bus residing on a CardBus PC Card;
- CardBus as a docking link between a host PCI bus and a “dock-side” higher-level PCI bus; and
- CardBus as a conduit for high-bandwidth video data flowing between a CardBus card and a host system’s PCI-resident main memory or video memory.

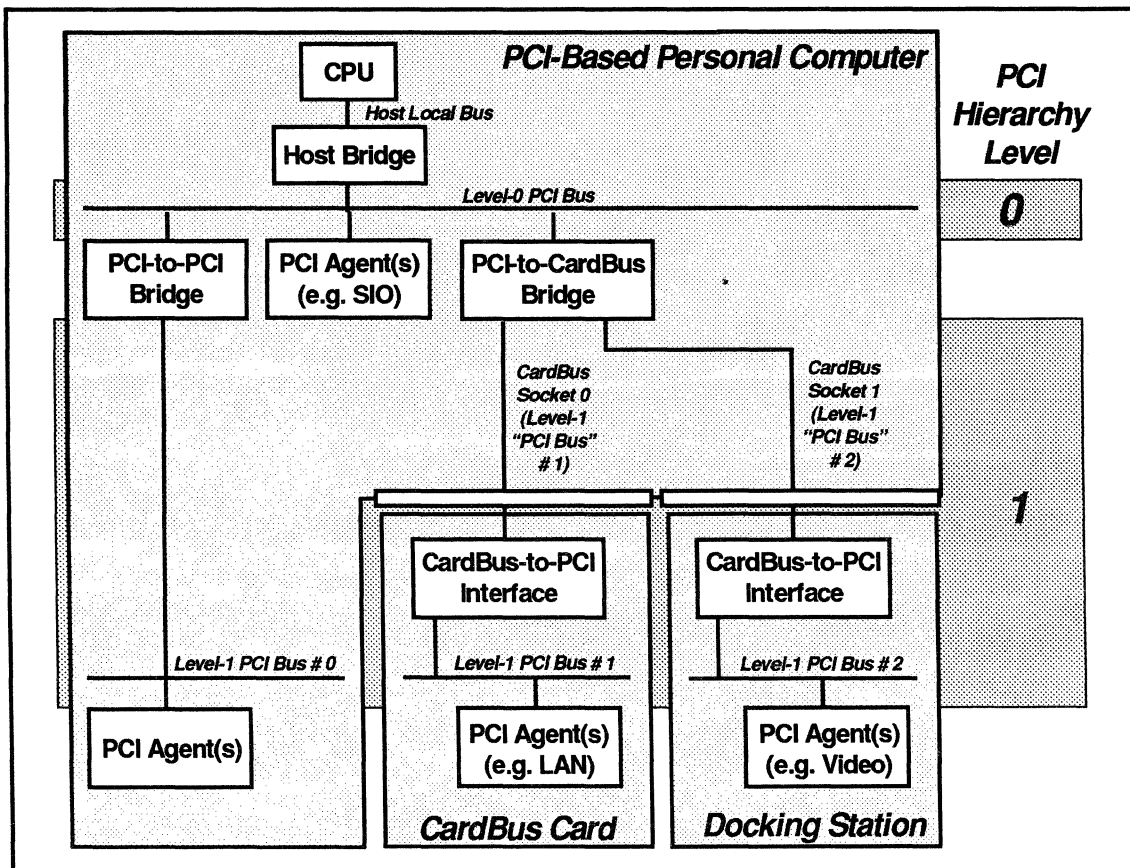


Figure 1: PCI Bus Hierarchy

## CardBus and PCI

Over the last two years or so, the Personal Computer Memory Card Industry Association (PCMCIA) standards body has developed an interface which extends the popular PC Card add-in standard. This "CardBus" interface extends both the performance and the functionality of the older "PC Card-16" interface. While the latter is ISA-like in its signaling and protocols, CardBus was deliberately designed to work seamlessly with the more recent PCI bus. Apart from electrical-environment differences, the similarity between CardBus and PCI is so marked that we may usefully think of CardBus as "point-to-point hot-insertable PCI" (see Figure 2). (By "hot insertion" we mean the ability to insert a PC Card into an operating platform, or to remove the card, without disrupting system operation).

	PCI	CardBus
<b>PERFORMANCE</b>		
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
<b>CONFIGURATION</b>		
Hot-Insertion Support	NO	YES
Boot-Up Configuration Support	YES	YES
Dynamic (Run-Time) Configuration	NO	YES
Configuration-Software Level	Low (BIOS)	High (Card/Socket Services)
<b>POWER MANAGEMENT</b>		
Operating Voltage(s)	5v/3.3v	3.3v Only
Card-Clocking Hardware Support	YES (PCI Mobile Only)	YES (CCLKRUN)
<b>MECHANICAL DESIGN</b>		
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 2: Comparison of CardBus and PCI

CardBus retains all of the major attributes of PCI--- particularly its synchronous nature, multiplexed address/data lines, multi-master capability, local-bus performance levels (up to 33 MHz operations at 32-bit data/address width), joint master/target transaction control, and integrated system resource-configuration capability. While CardBus is not restricted to usage in PCI-based systems, it is there that it especially shines.

Systems are now beginning to implement the hierarchical bus capability which PCI offers. This is especially true of high-end systems such as servers, in which higher-level PCI busses are needed to support high-bandwidth I/O activity and/or to allow overlapped activity on multiple busses. This hierarchical capability also allows systems to accommodate more PCI agents than the half-dozen or so which PCI's electrical loading rules allow on any one bus.

The constituent busses of such a hierarchy can be connected to or isolated from one another through "PCI-to-PCI bridge" devices. From the programming perspective, these bridges allow portions of a system's memory- and I/O address spaces to be mapped onto the host processor's (flat) memory and I/O spaces. System configuration software accomplishes this by programming address-space "windows" in the bridge hardware with the upper and lower address limits of each address block. This configuration is normally accomplished at POST time, though PC Card-equipped systems must be able to do this repeatedly as PC Cards are inserted into or removed from the host platform.

A PCI-to-CardBus bridge performs exactly the same task as is described above, in order to map PC Card-resident resources into the host system. With minor differences (e.g. a 4-Byte I/O-window resolution and granularity, vs. PCI's 4-Kbyte resolution and granularity), a CardBus bridge simply acts as one of the inter-bus gateways in a hierarchical PCI system. Such a device typically supports two CardBus sockets, effectively adding two independent "branches" to the system bus "tree". Note that a system may include several such bridges, and that the PC Card standard requires each CardBus socket to support both CardBus cards and PC Card-16 cards.



The major difference between a PCI-to-CardBus bridge and a PCI-to-PCI bridge is that the latter implements a full normal PCI bus on its secondary (“downstream”) interface, while a CardBus bridge is limited to a single downstream load. (The PCMCIA committee is in the process of relaxing this requirement somewhat to allow a single additional “stub” connection, as we will discuss later). In spite of this electrical loading limitation, CardBus makes provisions for “multi-function” PC Cards, in which several distinct functions (analogous to PCI agents) can share a single CardBus card-side interface.

Host-generated PCI Configuration bus cycles can be targeted at specific on-card functions (each of which has its own set of Configuration registers), just as they can with any PCI agent. These function Configuration registers include base-address registers which can be used to assign each function one or more sub-portions of the address blocks mapped by the CardBus bridge windows. Each CardBus-card function’s Configuration registers also include a pointer to a standardized set of “CardBus function” registers which are used to control and communicate with that function (e.g. to support PC Card insertion and removal notification, remote “wake-up” events, etc.).

System-level power management is taking on ever-increasing importance in computing platforms, and especially in mobile systems (for which extended battery life is a requisite). The “PCI Mobile” standard defines a “CLKRUN#” signal and associated protocol, through which a PCI bus clock can be turned off and on as needed to conserve dynamic-switching power; the system CPU and individual PCI agents on the bus negotiate for control of the clocking. The CardBus standard includes a “CCLKRUN#” mechanism which is patterned on the CLKRUN# protocol. Using this, systems which implement CCLKRUN# can extend power management to agents which reside on CardBus cards. PCI and CardBus thus share this important power-management mechanism, which can be implemented throughout a bus hierarchy.

### ***Continuing the Bus Hierarchy onto CardBus Cards***

We have seen how a PCI-to-CardBus bridge maps PC Card-resident resources onto the host’s address spaces, and how a host can configure multiple card functions via PCI Configuration bus cycles. In all of this, PC Card functions behave just like PCI agents located downstream of a PCI-to-PCI bridge. Each such function can claim bus cycles within its programmed address windows. Conceptually, the card’s CardBus interface “fans out” CardBus transactions to all of the card’s functions, as if they resided on a local (PCI) bus.

A PC Card CardBus interface must satisfy CardBus loading requirements, even though there may be multiple functions on the card. This means that it may be necessary to buffer the card’s CardBus interface en route to the several functions. The card’s CardBus interface must also combine the function-interrupt lines from the various functions, to drive the interface’s single CINT line. (Since the interrupt line is shared by all functions on the card, software must poll for the source of an active interrupt. This can be done in accordance with the existing PC Card multi-function interrupt-sharing protocol.) Similarly, the card’s CardBus interface must combine the card status-change line from all functions to drive a single interface CSTSCHG interface signal.

The system configuration mechanisms of PCI and CardBus are essentially identical, as shown in Figure 3. Both busses support “Type-0” Configuration bus-cycles, which are used to configure agents which reside on the bus which receives the Type-0 cycle. The upper 21 address bits of a Type-0 cycle are used to select a particular device (i.e. a PCI agent on a PCI bus, or a CardBus card on a CardBus interface). The lower address bits are used to direct Configuration cycles to a particular function (within a multi-function device), and to a particular Configuration register within that function.

Both PCI and CardBus also support “Type-1” Configuration bus-cycles, which can be relayed down the bus hierarchy to destinations which lie behind bridge devices. When a Type-1 Configuration cycle reaches its target bus, it is converted to a Type-0 cycle, which is then processed as previously described. This allows devices to be found, classified and configured anywhere within the bus hierarchy. A particular usage of this mechanism also allows PCI “Special” cycles to be sent to destinations throughout the bus hierarchy. Special cycles can be used in lieu of dedicated special-purpose hardware signals, to perform tasks like information broadcasts.

Throughout the hierarchy, PCI-to-PCI bridges and PCI-to-CardBus bridges include Configuration registers which are used to assign unique numbers to each system bus. Bridges also contain other Configuration registers which specify what range of bus numbers lie behind each particular bridge. A piece of system software called a “bus enumerator” is used to catalog system resources and program the preceding Configuration registers; this usually is done at POST time, but must be redone as PC Cards are inserted into or removed from the system.

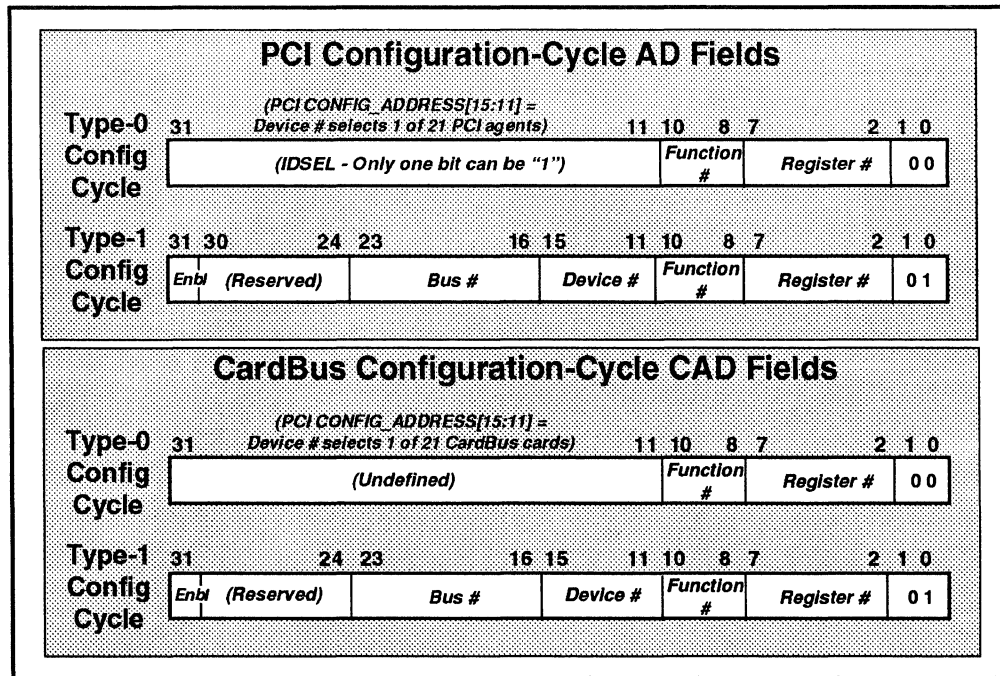


Figure 3: PCI and CardBus Configuration Addressing

In early CardBus cards, the card's functions may be viewed as terminal "leaves" of that branch of the system bus tree which lies downstream of a PCI-to-CardBus bridge. However, the bridge's PCI header includes a subordinate-bus Configuration register similar to that found on PCI-to-PCI bridges. This register can be programmed so as to indicate that multiple busses lie downstream of the bridge. It should be possible to build a card-resident CardBus-to-PCI interface, such that the card contains a local full PCI bus which can be populated with normal PCI agents; this would effectively continue the system's bus hierarchy onto the CardBus card. Since this is not a practical alternative in the near-term, we will not explore it further here.

#### **Linking Platform and Dock Busses via CardBus**

At present, specialized PCI-to-PCI bridges are being used as a mechanism for linking ("docking") a portable computer to a PCI-based "docking station", as illustrated in Figure 4a. (Earlier docking approaches used ISA-based mechanisms). In this application, a PCI-to-PCI bridge provides the necessary address-mapping facilities, and an associated set of buffers are used to electrically connect the bridge's "downstream" interface to the docking station. This electrical connection/isolation capability is referred to as "hot insertion".

A PCI-to-CardBus bridge has inherent hot-insertion and dynamic configuration capabilities, making it an ideal candidate for docking applications (see Figure 4b). With a CardBus bridge, external isolation buffers are unnecessary, and existing CardBus system software (Card and Socket Services) provides a means for dynamically managing dock-resident resources as docking and undocking occur. The dock's resources (PCI agents) can be identified, configured and used by host-resident system software (e.g. a PCI bus enumerator), with the CardBus docking connection serving as a link in the combined host/dock PCI bus hierarchy.

Using a CardBus bridge as a docking medium entails dealing with essentially the same issues as using such a bridge to continue the system bus hierarchy onto a card-resident PCI bus. In this case, though, the downstream bus resides in a docking station, rather than on a CardBus card. The dock can be fitted with a CardBus interface which can be used to generate functional interrupts and status-change signals to the mobile-platform processor. The dock's PCI bus can support PCI agents such as video controllers, storage-media controllers, and various connectivity adapters. In addition, if required, a PCI-to-ISA bridge can be added to the dock's PCI bus, to support legacy hardware and software. (Note that the new industry-standard "PCIway" serialized interrupt and distributed DMA mechanisms provide purely PCI-based means to support ISA legacy functions, potentially eliminating the need for actual ISA hardware).

This docking approach uses standardized CardBus hardware and software mechanisms to support dynamic system reconfiguration following docking and undocking; moreover, it does so in a way that is fully consistent with the

system's PCI bus hierarchy. The host's CardBus socket controller fulfills the same functions as three separate blocks in a PCI-to-PCI docking interface: isolation buffers, PCI bridge and status-change generator. These benefits make CardBus-based docking of PCI platforms and docks more attractive than today's more ad-hoc approaches.

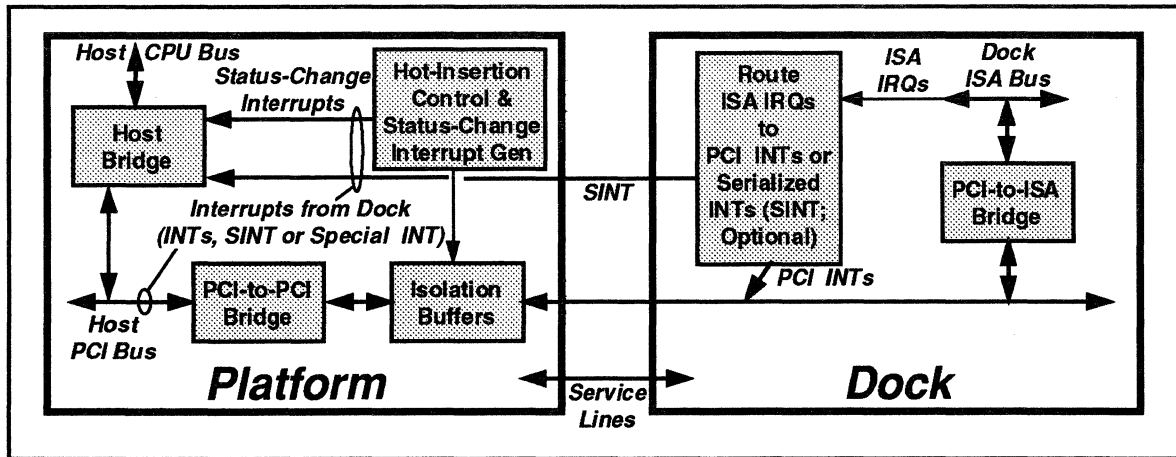


Figure 4a: Docking via PCI-to-PCI Bridge

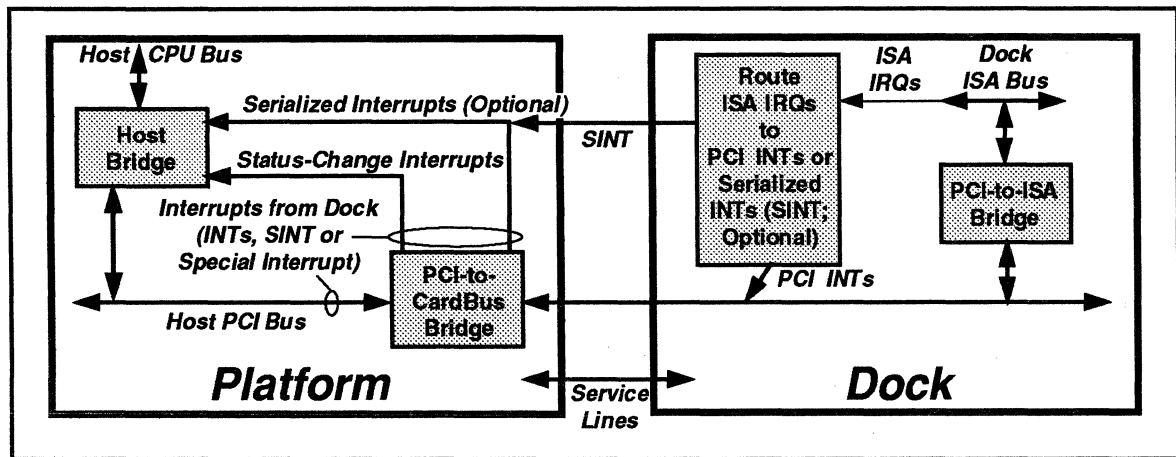


Figure 4b: Docking via PCI-to-CardBus Bridge

Perhaps the biggest issues with this style of docking center on mechanical engineering issues, rather than on docking functionality problems. For this approach to support standardized docking across platforms and docking stations, manufacturers would have to agree on placement of the CardBus docking socket, as well as on how to handle any remaining non-CardBus "side-band" lines, such as serialized-interrupts (SIRQ) and service connections. (Use of a normal CardBus socket and connector are assumed, since the CardBus electrical specifications probably cannot be met using a connector cable). In addition, system manufacturers would need to see benefits to them in adopting a standardized docking approach, which would decrease the value of proprietary docking solutions. Still, the potential flexibility and cost-reduction benefits to the end-user are clear.

### CardBus and PCI Multi-Media Busses

CardBus is a useful adjunct in multi-media-capable PCI systems (see Figure 5). CardBus cards can be used to add or enhance video and audio capabilities to a system. As an example, a video "front-end" may be implemented on a CardBus card. In this arrangement, a PCI-to-CardBus bridge must provide a bandwidth-efficient video "gateway" from the Card onto the host's PCI bus(es). To prevent the video data from consuming excessive bandwidth on the host's level-0 PCI bus (potentially on Paths A or B below), the CardBus bridge can direct video data onto a secondary host PCI bus (Path C), or onto a specialized video "side-band" path, as is done in Zoom Video (Path D). In this video application, CardBus serves as an important data-routing element within a system's overall bus hierarchy.

Multi-media applications can consume a considerable amount of bus bandwidth. If high-resolution, real-time video data flows over a system's primary PCI bus, it can detract from bandwidth which is needed by the system CPU. Conversely, CPU utilization of the primary bus can interfere with video-subsystem performance by introducing excessive video-data transfer latency, or by leaving inadequate bandwidth for the video data.

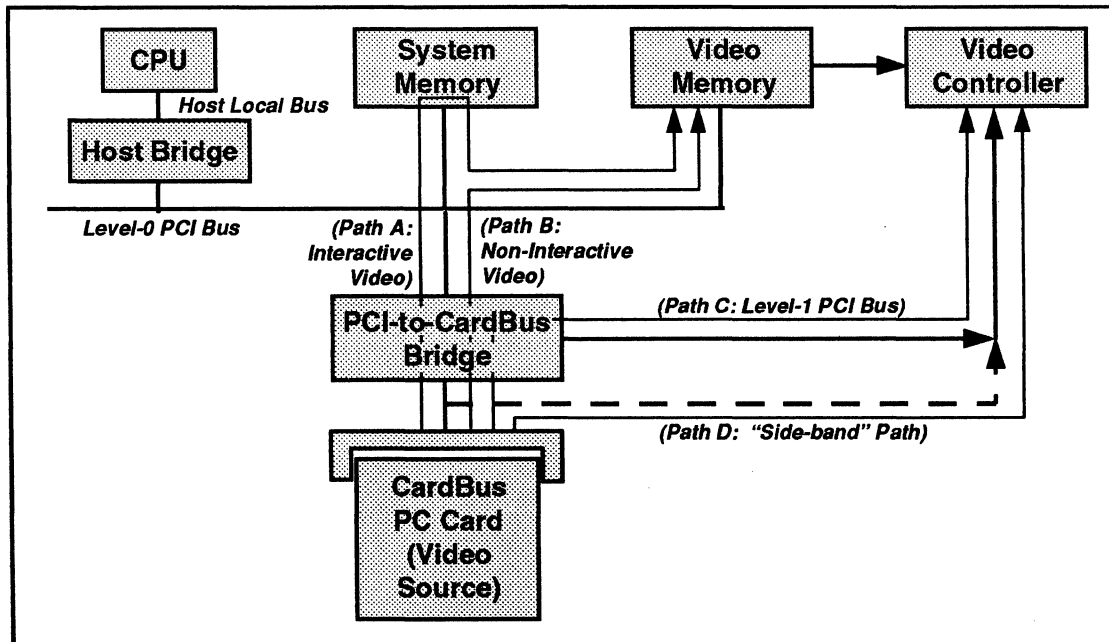


Figure 5: CardBus as a Multi-Media Bus

In a hierarchical PCI system, it is possible to direct high-bandwidth I/O or memory traffic over a secondary PCI bus, thus avoiding or greatly reducing the impact of this traffic on the primary bus. Such an arrangement is depicted for paths C and D in Figure 4. In path C, a PCI-to-CardBus bridge can be used to direct the video data onto a secondary PCI bus which is connected to the system's video controller. This requires a PCI-to-CardBus host-side controller which supports two distinct PCI interfaces, as well as CardBus sockets. In path D, the video flows over a "side-band" path which circumvents the bridge altogether; the bridge simply provides bus isolation to keep the video data apart from the rest of the bus hierarchy. In either arrangement, the video data is kept from interfering with the system's primary bus.

Note that multi-media applications present special problems for system bus design. In particular, both audio data and video data demand that certain timing constraints be met, or audio/video performance can be compromised (e.g. video "tearing", choppy audio, etc.). Meeting these timings requires careful analysis of the system busses, as well as appropriate design of bus arbiters and device buffers. This problem is substantially easier to solve on a dedicated multi-media secondary PCI bus, rather than on the overall system bus.

PCI devices include required Configuration registers which can be used to tune device timing characteristics, such as maximum bus-acquisition latency, minimum tenure as master, and minimum acceptable bus-acquisition frequency. CardBus Configuration registers provide these same capabilities. Bus arbiters can be designed to devote a given fraction of total bandwidth to particular devices, so that they can equitably share a bus. Jointly, these mechanisms provide designers a relatively high (though not absolute) measure of control over bus utilization. However, proper usage of these mechanisms is application-dependent, and may be crudely supported by system software (e.g. BIOS).

### **Software Support for the Bus Hierarchy**

As we have seen, there is much hardware synergy between CardBus and PCI. To take advantage of this, though, applications must be supported by adequate system software. Today's system software cannot yet completely provide this support.

As mentioned earlier, a PCI BIOS includes a bus enumerator which is used to find all PCI devices in a system, as well as all PCI busses in a multi-bus hierarchical system. The enumerator assigns unique numbers to all busses, and

writes these numbers to the primary-bus Configuration register of each such device. For bridge devices, such as PCI-to-PCI bridges and PCI-to-CardBus bridges, the enumerator also writes the assigned bus numbers to each bridge's Secondary Bus Number and Subordinate Bus Number Configuration registers. This device configuration activity captures a particular system's topology (connectivity pattern), and makes it available to software.

In a PCI system without CardBus, the bus enumerator is invoked once after system boot-up, normally at POST (power-on self-test) time. This is consistent with the fact that the resources in such a system do not change over time. The situation is much more complex when the system can change due to CardBus card insertion or removal. In that case, the bus-enumeration process has to be repeated with each card insertion or removal. Doing this involves adding mechanisms for detecting card events; this is the purpose of the CardBus (and PC Card-16) "status-change" signal.

PC Card software supports dynamic system configuration. "Socket Services" is a hardware-*dependent* software layer which operates by making calls on lower-level BIOS functions. (Note that BIOS is tailored for a particular system). Socket Services allows a hardware-*independent* "Card Services" layer to manipulate a particular PC Card bridge and a particular type of PC Card in a standardized manner, by making calls on Socket Services functions. Some Card Services functions can be used to ascertain the resources needed by a particular PC Card, such as interrupt level, Vcc and Vpp voltages and currents, DMA channels, etc. This is done when a PC Card "registers" itself with Card Services after card insertion. Other Card Services functions can be used to allocate and free such resources for use by PC Cards.

The boundaries between BIOS, OS and PC Card software are changing. The trend, which is driven by Microsoft, appears to be toward integration of BIOS functions into an OS "hardware Abstraction Layer" (or "HAL"), as well as absorption of PC Card functions into the OS itself. This trend is not yet complete; for example, Windows 95 does not include native support for CardBus (though it remains compatible with various implementations of CardBus-capable Card and Socket Services. The direction appears clear, though. Over time, operating systems promise to provide uniform support for the various elements of a PCI hierarchy, be they PCI agents or CardBus cards.

### **Summary**

CardBus is a natural complement to PCI in implementing a hierarchical PCI bus structure. CardBus and PCI share the same system resource-configuration mechanism, and these busses are well-matched in terms of performance. PCI-to-PCI bridges allow a set of PCI busses to be connected in a *static* (i.e. hard-wired) topology, so that these busses can subsequently be either connected to or isolated from one another under program control. PCI-to-CardBus bridges allow portions of a bus hierarchy to be *dynamically* added to or removed from the system, either on CardBus cards or on a PCI-based docking station.

The CardBus hardware standard supports this through a "hot-insertion" capability, automatic card-type determination, socket status-change and interrupt mechanisms, and a robust connector and card form-factor definition. In addition, the CardBus software standard (which consists of "Metaformat", "Socket Services" and "Card Services") prescribes a method for handling dynamic resource configuration and management. Taken together, PCI and CardBus constitute a unified solution to the needs of hierarchical PCI-based systems.

---

### **Author's Biography**

Claude Cruz is a PCMCIA Product Architect with National Semiconductor Corporation in South Portland, Maine, where he focuses on CardBus-related products. Mr. Cruz has extensive digital systems design experience, including 11 years as an IBM designer and technologist, and 7 years as a consultant in parallel-processing and DSP systems, neural networks, fuzzy logic and AI. He holds a double BS in Electrical and Biomedical Engineering from the University of Southern California, and a joint MS in these same fields from the University of Illinois at Champaign-Urbana.

**WHERE DO I PLUG THE CABLE?  
SOLVING THE LOGICAL-PHYSICAL SLOT NUMBERING PROBLEM**

**Jeff Autor and Alan Goodrum  
Compaq Computer Corporation  
PO Box 692000  
Houston TX 77269-2000**

**jautor@bangate.compaq.com      agoodrum@bangate.compaq.com**

**ABSTRACT**

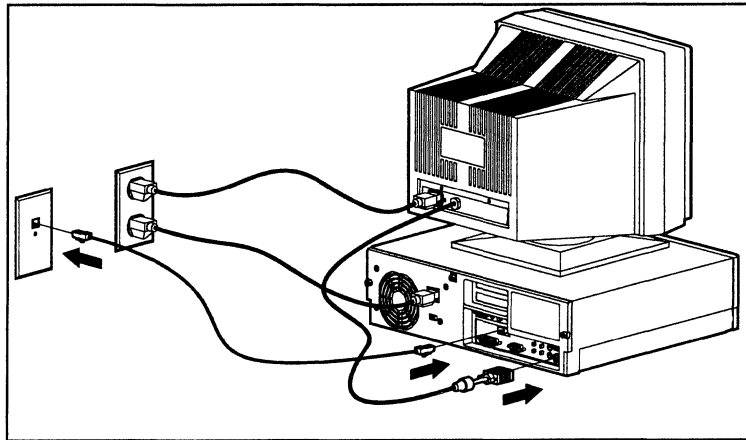
As the number of identical PCI devices performing unique functions in one server increases, it becomes increasingly difficult to physically identify a specific PCI device. This paper will explain the need for communicating to the user the unique physical location of a specific PCI device, specifically the chassis and slot numbers. New PCI-to-PCI bridge registers designed to help solve this problem and defined in the upcoming revision to the PCI-to-PCI Bridge Architecture Specification are described. The algorithm for a proposed PCI BIOS call is also presented. The new BIOS call uses the new bridge registers to convert between logical bus and device number and physical chassis and slot number.

**THE NEED FOR SLOT NUMBERS**

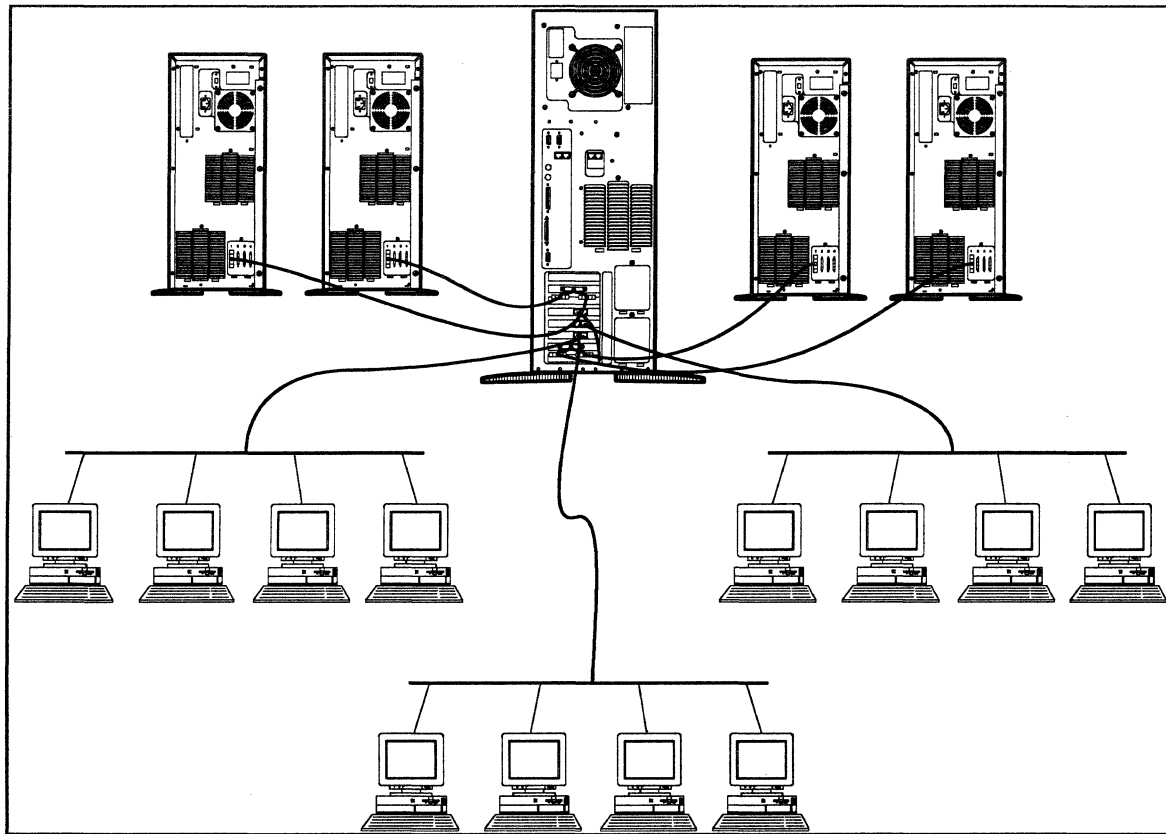
The PCI standard has been able to deliver on the “plug and play” promise by requiring that any compliant device be able to accept any valid resource configuration at power-up. PCI BIOS assigns resources at power-up, automatically allocating system resources without conflict. However, unlike previous standards, the PCI standard does not include the concept of a “slot,” that is, a physical geographic description of a device’s location within the system.

**Desktop Computers**

In a standard desktop computer, there are usually few PCI expansion slots and rarely multiple instances of the same device, making it easy to identify physically any particular device. The typical desktop includes slots for a graphic controller, a network controller, and mass storage controller. There is one connector for each, and using shape and size alone, the cables can be successfully attached to proper devices. With PCI the configuration process executes each time the machine powers on, so resource conflicts do not occur, even if a controller has been exchanged or a new controller added.



**Figure 1—Desktop computer applications typically have few external connections and no duplicate connectors, simplifying the connection process. This one has only a video monitor and LAN.**



**Figure 2—A typical server (center) may have identical electrical connections to multiple storage subsystems (across the top), and multiple identical electrical connections to different LAN segments (bottom).**

### Servers

Identifying a particular physical device becomes confusing with PCI-based network servers. A PCI-based server typically contains a large number of PCI expansion slots, averaging six to eight slots by mid-1996. These expansion slots are likely to be filled with multiple, sometimes identical controller-types that provide support for network segments, and multiple disk channels. Servers containing four or more disk controllers or five network controllers are not uncommon, especially in large database configurations. While one disk controller may connect to a number of SCSI disks, another may control multiple tape drives. One network controller may connect to hundreds of systems in an office building, while another network controller handles a connection to the Internet.

With PCI, the user no longer needs to manually allocate interrupts or memory address ranges. However, there are still situations in which a user must identify a particular controller both logically and physically. For example:

1. When plugging in an external cable, the user must identify the correct connector.
2. When configuring items such as the operating system, device drivers, and protocol stacks, the software will require a way to identify the device. For example, when configuring network controllers, the user must typically specify a controller (identified by slot number), and then assign a network address and the protocols to use with that controller.
3. When a controller of any type fails in a system, software such as diagnostic tools must have a way to communicate which controller has failed so that the user can physically replace it.

Consider the following example: A PCI server with two identical Ethernet controllers has one controller cabled to a small number of workstations. The other controller is cabled to an Ethernet backbone that runs throughout the company. To properly configure all the software running on the server, the two network controllers must be assigned TCP/IP addresses. The user must match a software configuration parameter (the IP address) to a piece of hardware (one of the two controllers). Without a unique identifier such as a slot number, the user has no constant identifier that is guaranteed to remain the same no matter how the rest of the system is reconfigured.

The system software, of course, can uniquely identify each controller *logically*, by a PCI bus number and device number. The problem is presenting this information to the user, so that the user can *physically* locate the controller.

### Why Can't the User Use Bus and Device Number?

Although the PCI bus number and device number do uniquely identify each controller, this identifier falls short of the user's needs in two areas. First, the slot number is a familiar paradigm for users. Users already understand the concept of "slot number." Instructing a user to install a controller "at bus 0, device 4" would require a shift in the user's thought process. The slot number provides an intuitive method for the user to physically identify a controller.

But more importantly, using PCI bus numbers and device numbers as an identification method is deficient for another reason: PCI bus numbers do not necessarily remain constant. When multiple PCI host bridges, or PCI-to-PCI bridges are embedded in the system, there are multiple buses to enumerate, and these numbers can change when the system is reconfigured. Because bus numbers are assigned during the boot process, just like other system resources, there is no guarantee that they will remain constant across boot cycles. Thus, if the user configures software to use a controller found on "bus number 2, device number 7," and later adds another controller that happens to have its own PCI bus embedded, then any bus number beyond bus 0 will potentially be reassigned. The reassignments are based on the location of the controller, and in what order the system's BIOS finds and configures PCI devices during the boot process. If bus number 2 is reassigned to bus number 3, then the user's software configuration would be incorrect, as would any slot markings or configuration notes he may have made to help locate the device. A physical identifier, such as a slot number, remains constant across boot cycles, and therefore provides a better solution to the problem.

### SLOT NUMBERS IN THE IRQ ROUTING TABLE

These types of challenges brought about the creation of the IRQ Routing Table call in the PCI BIOS, which was added to the *PCI BIOS Specification*, Revision 2.1. Using the bus number and device number, software can perform a table lookup to retrieve information about how each device in the main chassis is wired. One of the

Offset	Size	Field Description	Value
0	byte	PCI Bus Number	00h
1	byte	PCI Device Number (in upper 5 bits)	58h
2	byte	Link value for INTA#	3
3	word	IRQ bit-map for INTA#	0FFFFh
5	byte	Link value for INTB#	4
6	word	IRQ bit-map for INTB#	0FFFFh
8	byte	Link value for INTC#	3
9	word	IRQ bit-map for INTC#	0FFFFh
11	byte	Link value for INTD#	4
12	word	IRQ bit-map for INTD#	0FFFFh
14	byte	<b>Physical Slot Number</b>	<b>5</b>
15	byte	Reserved	0
16	byte	PCI Bus Number	00h
17	byte	PCI Device Number (in upper 5 bits)	70h
18	byte	Link value for INTA#	5
19	word	IRQ bit-map for INTA#	0FFFFh
21	byte	Link value for INTB#	6
22	word	IRQ bit-map for INTB#	0FFFFh
24	byte	Link value for INTC#	5
25	word	IRQ bit-map for INTC#	0FFFFh
27	byte	Link value for INTD#	6
28	word	IRQ bit-map for INTD#	0FFFFh
30	byte	<b>Physical Slot Number</b>	<b>6</b>
31	byte	Reserved	0
32..xx		<i>Additional PCI Device Entries</i>	

Figure 3—An excerpt from a typical IRQ Routing Table defining how PCI interrupts are connected for devices in the main chassis. This table can be used to translate between PCI bus and device number, and slot number for devices in the main chassis.

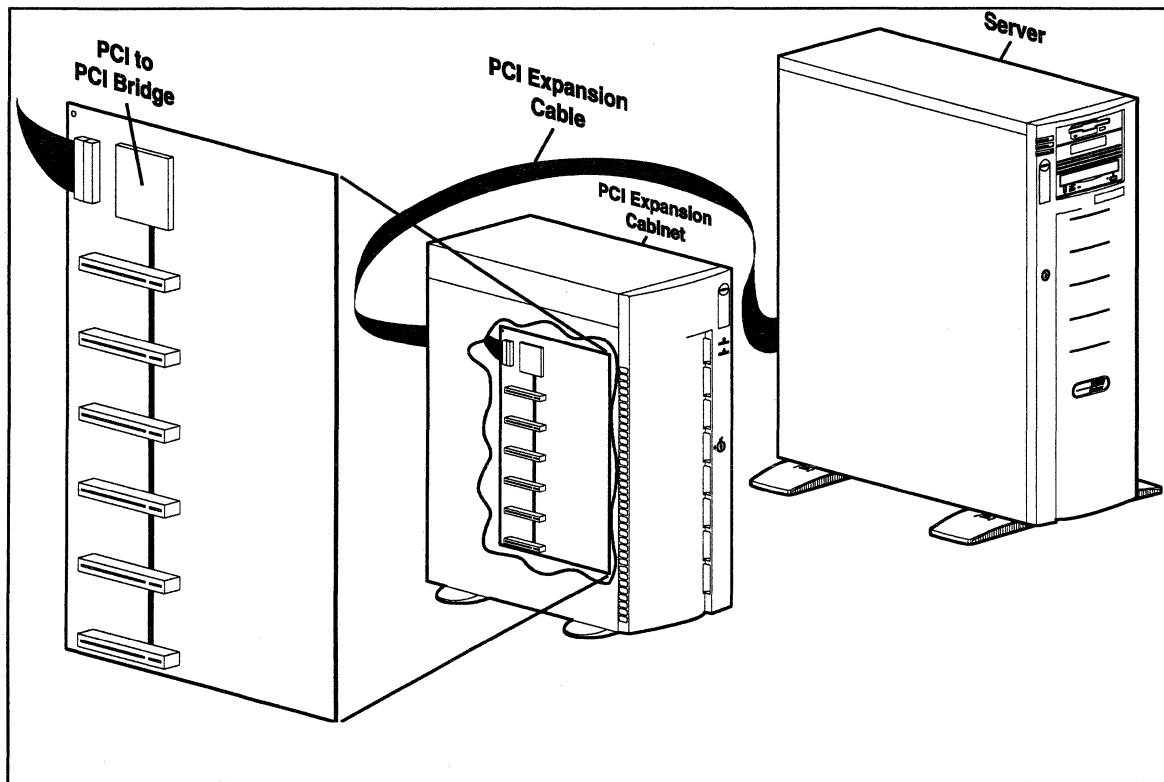


fields defined in the IRQ Routing Table is the device's slot number, as shown in Figure 3. The software uses the information in the IRQ Routing Table to translate the physical slot number into PCI bus number and device number for devices in the main chassis.

### **PCI EXPANSION SYSTEMS**

A PCI expansion system can be described as an external cabinet containing PCI expansion slots, which connects to a server through one or more PCI-to-PCI bridges, as shown in Figure 4. Expansion systems are a recent addition to the PCI product landscape, because they are only useful in server environments where large numbers of I/O controllers (i.e., disk controllers, network and communications controllers) are used. A network file server may require these expansion cabinets when all PCI slots in the server are already in use.

Expansion cabinets complicate the problem of physically locating a device. Not only does the user need to locate a connector in a specific slot, he must also search multiple external cabinets for the controller. Furthermore, slots in expansion cabinets cannot be included in the IRQ Routing Table because the BIOS has no way of determining what expansion system might be installed by the user.



**Figure 4—In a PCI expansion system additional PCI slots are provided in a separate cabinet, further complicating the problem of unique physical identification of a device. The slot numbering proposal assigns a unique “Chassis Number” to each cabinet.**

### **A COMPREHENSIVE SLOT NUMBERING PROPOSAL**

Since the IRQ Routing Table solves the slot numbering problem in the main chassis, what is required is a standard method for determining slot number in a PCI expansion system. In mid 1995, Compaq Computer Corporation began circulating for review within the PCI community a proposal for a general solution to this problem. The hardware required to support this proposal is being included in Revision 1.1 of the *PCI-to-PCI Bridge Architecture Specification*. At the time of this printing Revision 1.1 is nearing

the review process within the PCI-to-PCI Bridge Subcommittee. A standard BIOS call which uses the new hardware and the IRQ Routing Table is being proposed to the PCI BIOS Subcommittee as well. The following discussion presents the hardware aspects of the proposal, followed by the software aspects.

**The New Registers**

If we assume that the gateway to an expansion system is always a PCI-to-PCI bridge, then the logical place to define a standard solution to the slot numbering problem for expansion systems is the bridge. However, before a new standard feature could be added to the PCI-to-PCI bridge programming model, another problem had to be solved. The standard bridge Configuration Space Header was full, so additional space had to be reserved. As shown in Figure 5, configuration addresses F0h through FFh are defined by the proposal to provide additional standard configuration space. Bit 15 in the Bridge Control Register (3Eh) can be read to determine whether this additional space is supported. The two new registers shown in Figure 5, the Chassis Number register and the Expansion Slot register provide the necessary information to make the device number to slot number conversion.

31		16	15		8	7		0	
Reserved				Chassis Number		Expansion Slot			F0h
		Reserved							F4h
		Reserved							F8h
		Reserved							FCh

**Figure 5—The two newly defined registers for slot numbering are located in a newly defined extension to the standard Configuration Space Header for PCI-to-PCI bridges.**

Each cabinet in the system which contains PCI slots is assigned a unique chassis number, with the host system assigned chassis number 0. The new Chassis Number register in the PCI-to-PCI bridge contains a single 8-bit number that designates the chassis number in which the slots on the bridge’s secondary bus reside. Multiple PCI buses contained in the same chassis should be assigned the same chassis number.

The Chassis Number register can be initialized either by the power-up system configuration software or by hardware. If the register is to be initialized by software, then the register will be read-write, and can either be non-volatile or can be initialized to 0 at power-up. If software determines that the register is read-write and the value is 0, or equals another chassis’ number, then software will assign a new chassis number. If the register is initialized by hardware, then the register will be read-only, and the system designer must provide a means for the user to change the chassis number if there is a conflict.

7	6	5	4	3	2	1	0		
Reserved		Slots Follow Parent	Expansion Slots Provided						F0h

**Figure 6—Expansion Slot Register. The information encoded in this register includes the number of expansion slots provided directly behind this bridge, and the Slots Follow Parent bit that indicates whether multiple bridges with expansion slots are cascaded within one chassis.**

The details of the Expansion Slot Register are shown in Figure 6. Bits 4-0 of the Expansion Slot Provided field contains the binary encoded value of the number of expansion slots which are provided directly on the secondary bus of this bridge. If no expansion slots are implemented behind a particular bridge, then this register should be initialized to 0.

To understand how the Slots Follow Parent bit is used it is first necessary to consider how PCI expansion systems might be configured. Figure 7 illustrates one such system. The bridge that controls the first slot in the expansion chassis (Bridge A in Figure 7) is referred to as the “parent” bridge. Its Slots Follow Parent bit is set to 0 to indicate that it is the parent. Additional bridges whose slot numbers follow

the parent slots (Bridges B and C in Figure 7) are referred to as “child” bridges, and their Slots Follow Parent bits will be set to 1.

Because the Expansion Slot Register provides the power-up system configuration software with vital information about the physical arrangement of the system, this register must be initialized before the power-up system configuration software runs. This generally implies that the Expansion Slot register must be initialized by hardware. The means by which the system designer programs this information into the hardware is not specified, and is, therefore, left to the creativity of the bridge designer. The simplest approach would be to initialize the register contents with the state of certain device package pins at RST# time. However, more elaborate schemes involving shift registers or even serial EEPROMs could reduce pin count or provide more flexibility and convenience to the user at the cost of increased hardware complexity.

### **Finding Chassis and Slot Number**

Chassis numbers are established by the system configuration software each time the system is reconfigured. The main chassis is always chassis 0, and expansion chassis numbers are stored in the Chassis Number registers in the appropriate bridges. After the system has been initialized, any software needing the chassis number for a device can first check the IRQ Routing Table to determine whether the device is in chassis 0. If not, the software must then find the bridge whose secondary bus number matches the bus number of the device in question. If this bridge supports expansion slots, then the chassis number can be read directly from the Chassis Number register. If this bridge does not support expansion slots, i.e. it is an embedded bridge, then the chassis number is read from the bridge which supports the slot in which the embedded bridge is installed.

The slot number of a device in the main chassis can be found just as simply as chassis number by looking in the IRQ Routing Table. However, in an expansion chassis the slot number of a device must be calculated from the device number and Slots Provided Register. The following assumptions are made to calculate the slot number for a device in a PCI expansion system:

1. Slot numbers within each expansion chassis start at 1 and increment sequentially.
2. The PCI device number for each expansion slot starts at 1 and increments sequentially.
3. If an expansion system has multiple child bridges with the same parent bridge, then the child bridge with the lower slot numbers must also have the lower device number on the parent bus.

To calculate the slot number for a device in an expansion chassis the software must first find the bridge whose secondary bus number matches the device’s bus number. If the Slots Follow Parent bit is *not* set in this bridge (this is a parent bridge), then the slot number is equal to the PCI device number. If the Slots Follow Parent bit is set in this bridge (this is a child bridge), the software calculates the slot number by adding the following three numbers:

1. Device number for this device.
2. Value from the Expansion Slots Provided field from the *parent* bridge.
3. Value from the Expansion Slots Provided field from all *other child* bridges of this parent, whose device numbers are less than the device number of *this* child bridge.

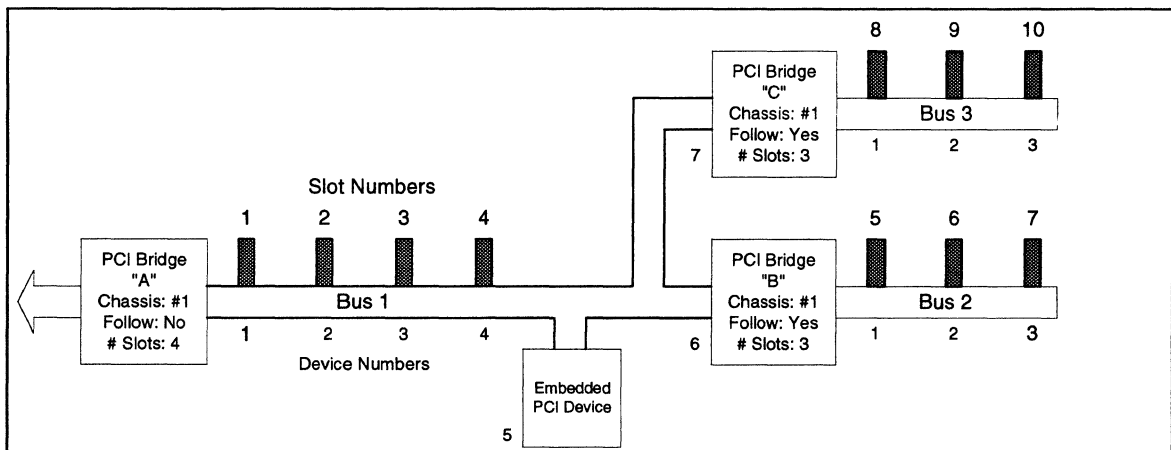
If the slot numbering algorithm encounters a bridge that does not support the Chassis Number and Expansion Slot Registers, then it is assumed that there are no expansion slots behind that bridge. All devices behind that bridge will inherit the same slot number as the bridge itself. In this way a card such as a multi-headed NIC or SCSI controller will report the same slot number for all devices on that card.

Figures 8 and 9 illustrate an algorithm of finding chassis and slot numbers for devices in an expansion chassis. The algorithm starts at the top of the configuration hierarchy and scans every device, accumulating chassis and slot information until the designated device is encountered.

### A Slot Numbering Example

The diagram shown in Figure 7 contains all the elements that can effect the numbering of PCI expansion slots. It represents a single, external expansion chassis, which would be connected to the system via the PCI-to-PCI bridge on the left side (the arrow indicates the connection to the system). Above each expansion slot is the Slot Number that would be physically labeled on the slot. The other numbers shown are the PCI Device Numbers that would be assigned to each (potential) device in the chassis.

The first PCI-to-PCI bridge (left side of the diagram) has four PCI expansion slots on its secondary interface (Bus 1). Since the Slots Follow Parent field (labeled “Follow” in the diagram) is not set, these slots must be the first slots within the chassis, and are therefore numbered 1 through 4. Also on Bus 1 is an embedded PCI device located at Device Number 5.



**Figure 7—PCI expansion chassis containing a hierarchy of bridges and devices. Bridge A is the “parent” since its slots come first and its Slots Follow Parent bit is reset. Bridges B and C are “children” since their slot number sequentially after Bridge A, and their Slots Follow Parent bit is set. Bridge B’s device number must come before Bridge C’s since Bridge B’s slots number first.**

At Device Number 6 is a PCI-to-PCI bridge, which reports three expansion slots on its secondary interface. This child bridge reports the same Chassis Number as the parent bridge, and its slots should follow those of the parent bridge. Since this bridge is the lowest numbered bridge device on Bus 1, its slots follow the parent bridge before higher-device numbered bridges. Therefore, its slots are numbered 5, 6, and 7.

The final device on Bus 1, Device Number 7, holds another PCI-to-PCI bridge, also reporting three expansion slots. Because its slots follow the parent bridge, the slots are numbered 8, 9, and 10.

### The “Find PCI Slot Number” BIOS Call

The slot numbering proposal includes the addition of a “Find PCI Slot Number” BIOS call to simplify the conversion between bus-device numbers and chassis-slot numbers for those operating systems that use the BIOS. Operating systems that do not use the BIOS will be able to do the same conversions by duplicating this algorithm within the operating system itself.

“Find PCI Slot Number” uses the IRQ Routing Table to find slot numbers in the main chassis (chassis 0), and uses the algorithm in Figures 8 and 9 for expansion chassis.

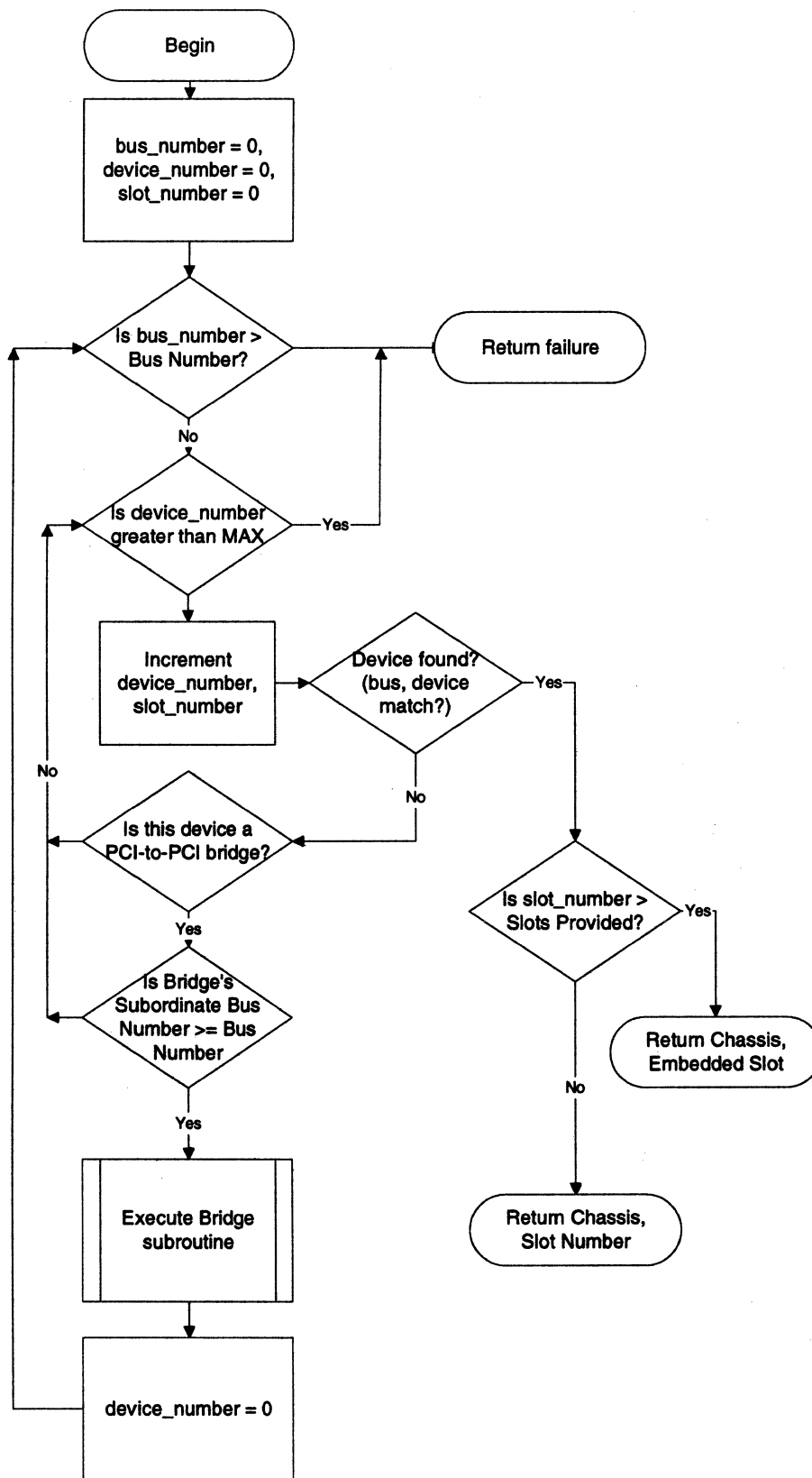


Figure 8—Flowchart for “Find PCI Slot Number” BIOS Function.

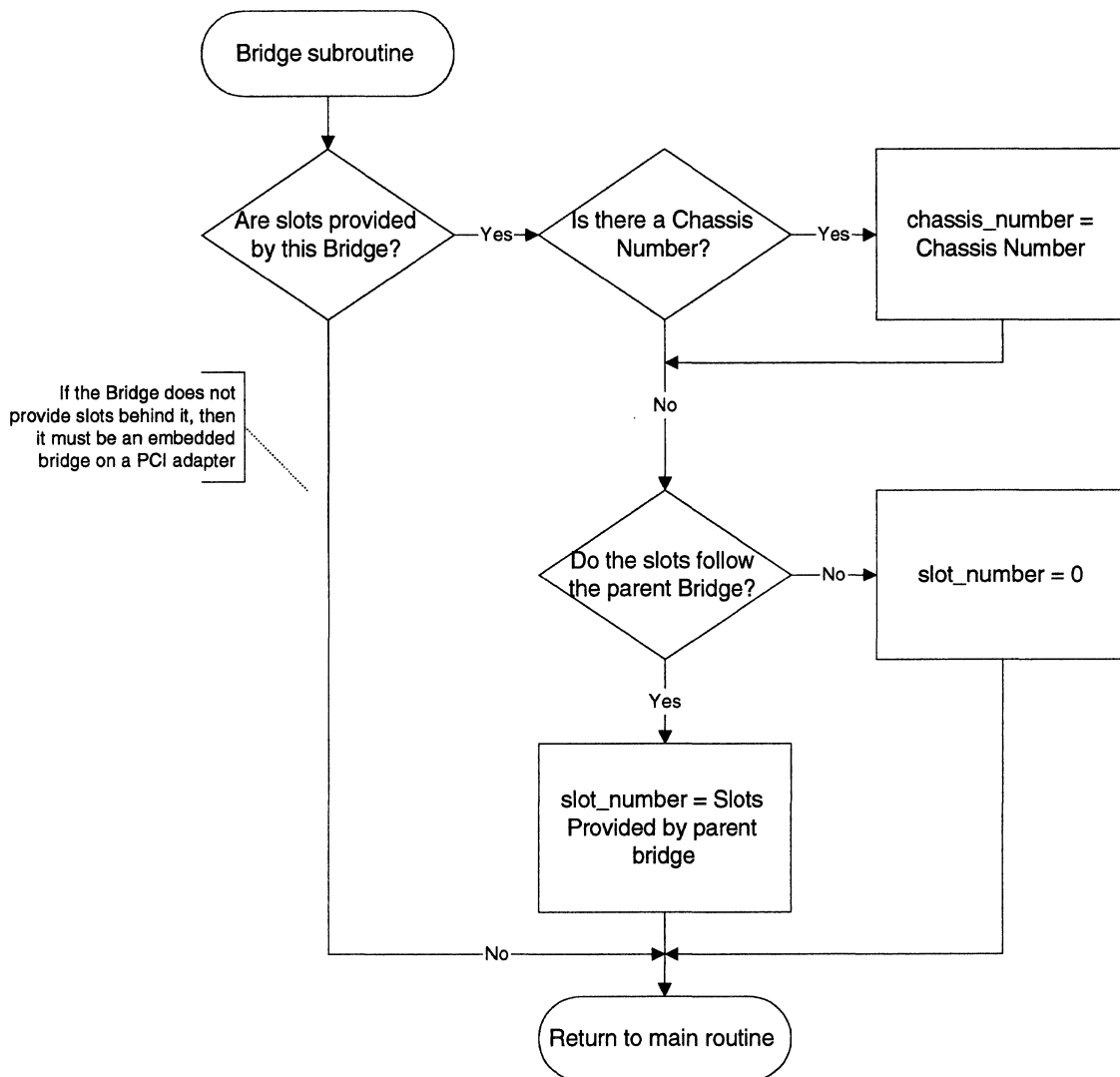


Figure 9—Flowchart for “Bridge Found” subroutine for the “Find PCI Slot Number” BIOS Function.

### The Future of PCI Slot Numbering

As mentioned previously, the hardware necessary to support this proposal in PCI expansion chassis is being added to version 1.1 of the *PCI-to-PCI Bridge Architecture Specification*. This same revision also specifies how Delayed Transactions work with PCI bridges. Compaq is encouraging multiple PCI-to-PCI bridge vendors to include the slot numbering register in their new Delayed Transaction bridge designs, even before the new revision of the bridge specification is released. Designers of PCI expansion chassis will naturally want to upgrade their products to take advantage of the performance gain of the new Delayed Transaction bridges as soon as they are available. When they do, we strongly encourage them to select a bridge that includes the hardware necessary to support PCI slot numbering.

Now that the hardware support is being implemented, support for a new BIOS call needs to be added. Compaq has already begun circulating a proposal within the user communities for review and comment. The proposal would add a “Find PCI Slot Number” in the next revision of the PCI BIOS Specification. Although the new hardware for slot numbering can be used without the new BIOS call, the inclusion of the new call will simplify the delivery of an accurate implementation of the algorithm, especially in areas such as diagnostics, system management utilities and BIOS-compatible advanced operating systems.

We expect that advanced network operating systems will not wait for the introduction of the hardware registers for the expansion chassis or for the BIOS call. These OS vendors will begin to implement the algorithm shown above immediately. Since slot numbers were added to the IRQ routing table in the August 1994 release of version 2.1 of the PCI BIOS Specification, solutions for the main chassis will already work with the current BIOS. When expansion chassis with the new bridge registers become available, numbering slots in the expansion chassis will work, too. Device driver and application writers should watch closely for developments from their OS vendor.

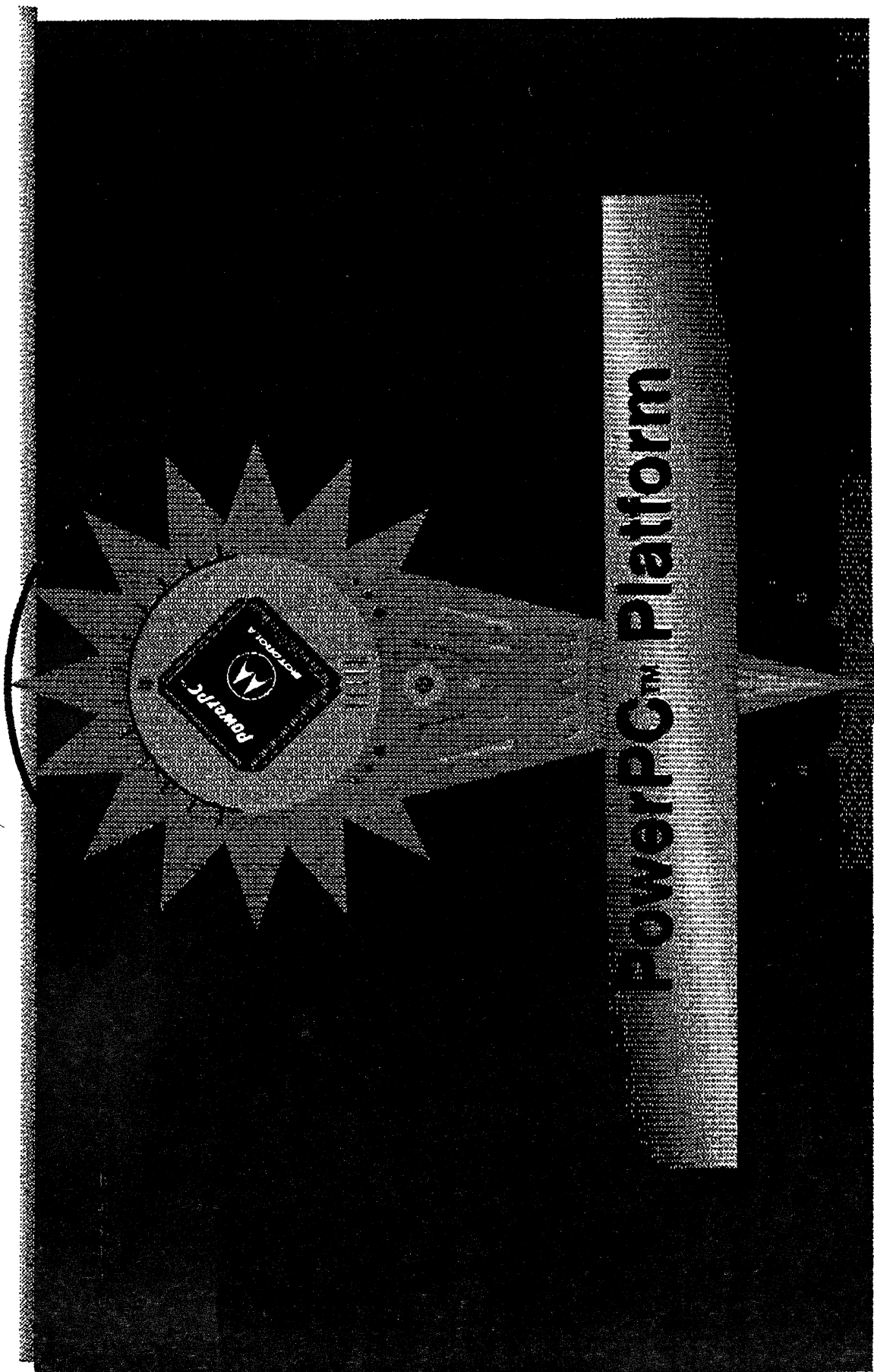
### **SUMMARY**

As the number of PCI slots grows to accommodate multiple identical controllers performing unique functions, it has become difficult to physically identify a particular controller. The inclusion of the slot number in the PCI BIOS IRQ Routing Table, and the Chassis Number and Expansion Slot Registers in new PCI-to-PCI bridge implementations will enable the translation from the logical bus and device number to the physical chassis and slot number for all controllers in the system.

### **THE AUTHORS**

Jeff Autor is a Systems Software Engineer at Compaq Computer Corporation. He has been designing and developing system management products for Compaq for the last five years. As part of the NetWare device driver development team, he co-designed Compaq's SNMP MIB, software for Compaq's Server Manager product, and most recently the Compaq ProLiant servers. He is currently working on systems management software for next-generation PCI-based server products. Jeff received a Bachelor's degree in Computer Engineering from the University of Illinois at Urbana-Champaign.

Alan Goodrum is a Principle Member of the Technical Staff in Compaq's Systems Division. He currently is involved with development of new technology which will be needed by file, print, and application servers several years in the future. Previously he was a senior architect in several hardware product development groups in the Systems Division since its creation in 1991, and a senior engineer at Compaq since 1985. Alan received his BSEE in 1975 and MSEE in 1984 from the University of Houston, where he has been a guest instructor.



# PowerPC™ Platform

**PowerPC™**

PowerPC Platform PRESENTATION - I

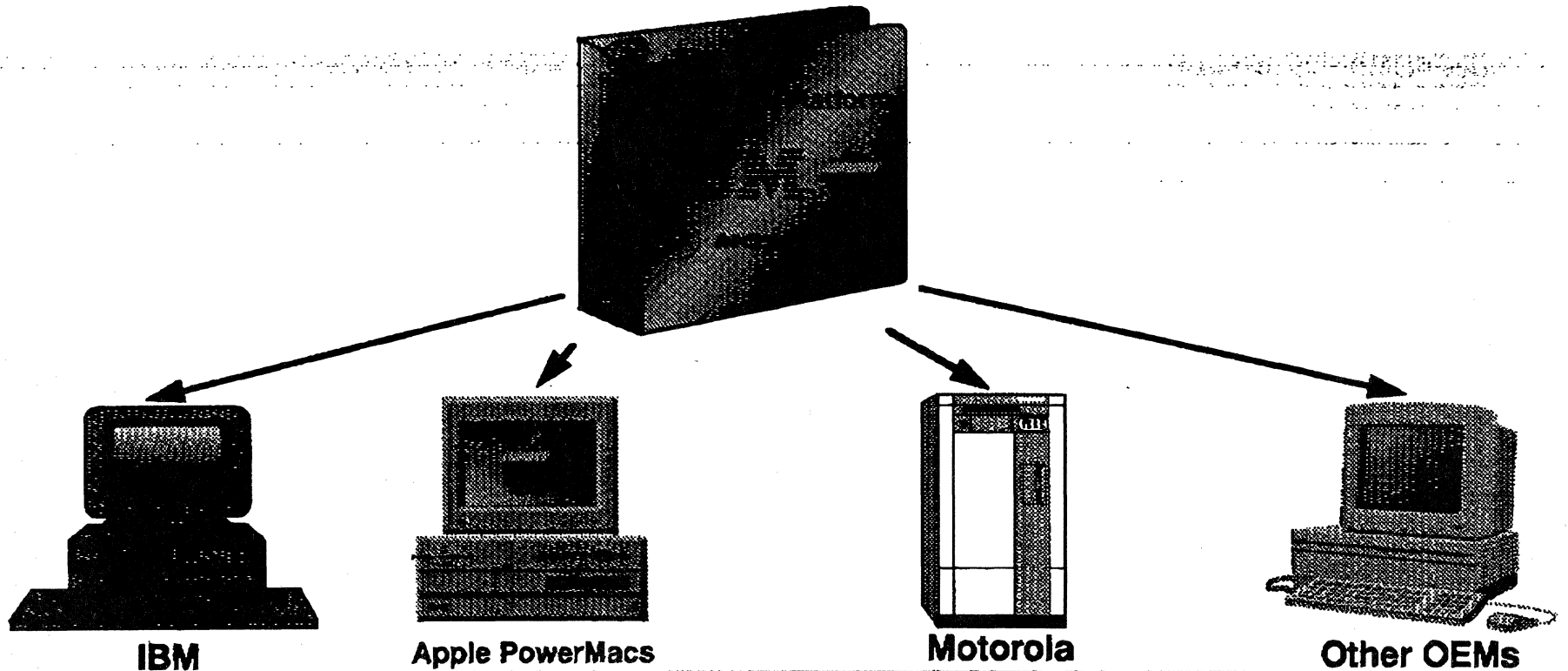


**MOTOROLA**



# What Is PowerPC Platform?

- ▼ PowerPC Platform supporting all PowerPC operating systems
- ▼ Open architecture built by any vendor with NO special licensing fees
- ▼ PowerPC Platform will run ALL applications developed for today's PowerPC systems including PowerMac systems



62

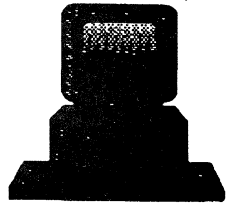
**PowerPC™**

PowerPC Platform PRESENTATION - 2

 **MOTOROLA**

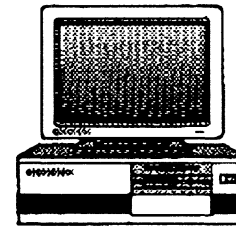
# Combining PowerPC Efforts

## PRP Implementations



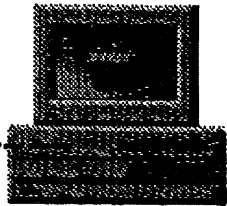
601 / 603 / 604  
Uni / MP  
PCI  
PC I/O  
ARC Firmware

## PowerPC Platform



63

## PowerMacs



601 / 603 / 604  
Proprietary logic  
Apple I/O  
NuBUS & PCI Bus  
Open Firmware

603 / 604  
Uni / MP  
PCI  
PC and Apple I/O  
Open Firmware

Total Application Software Compatibility Across All Platforms

1995

1996

**PowerPC™**

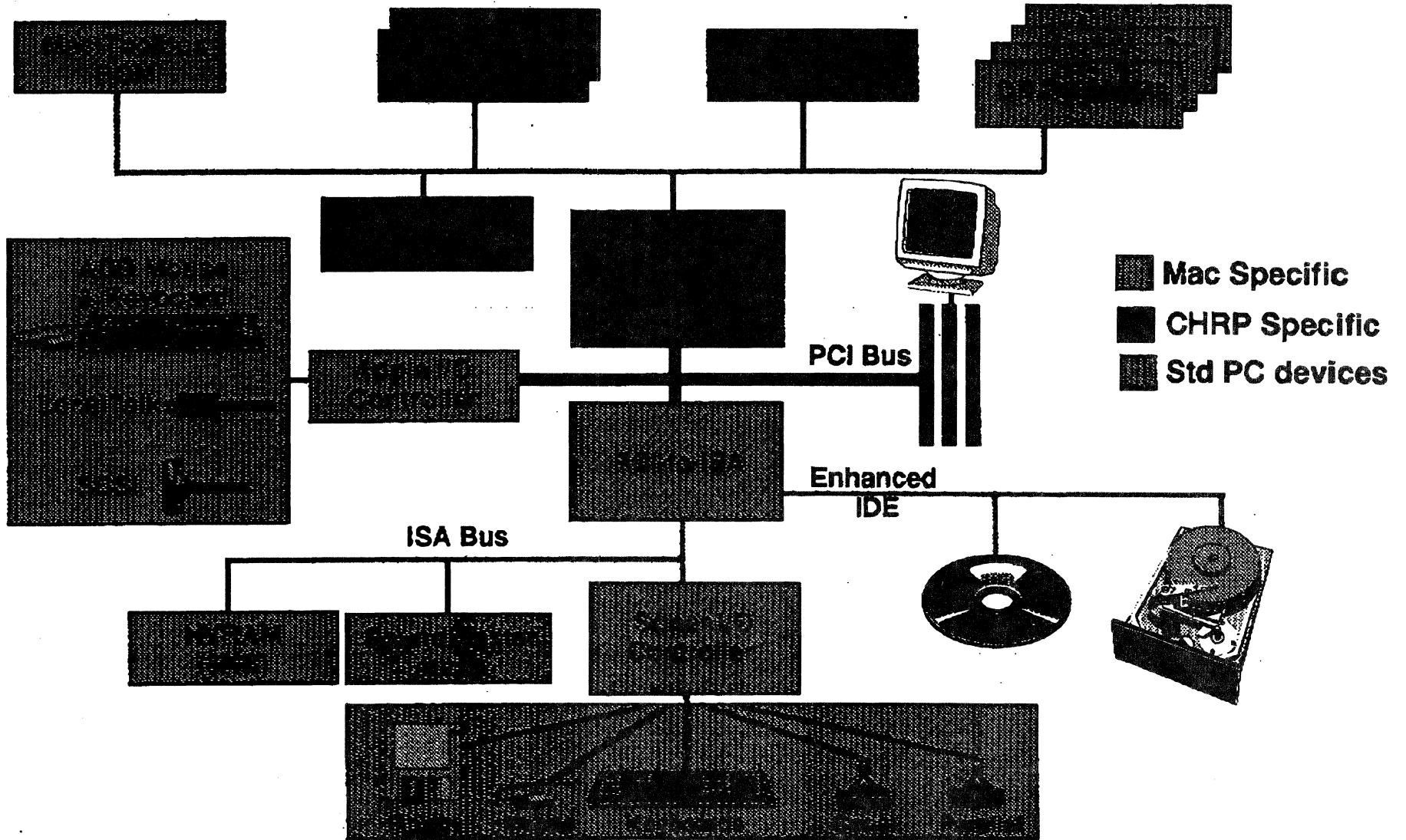
PowerPC Platform PRESENTATION - 3



**MOTOROLA**

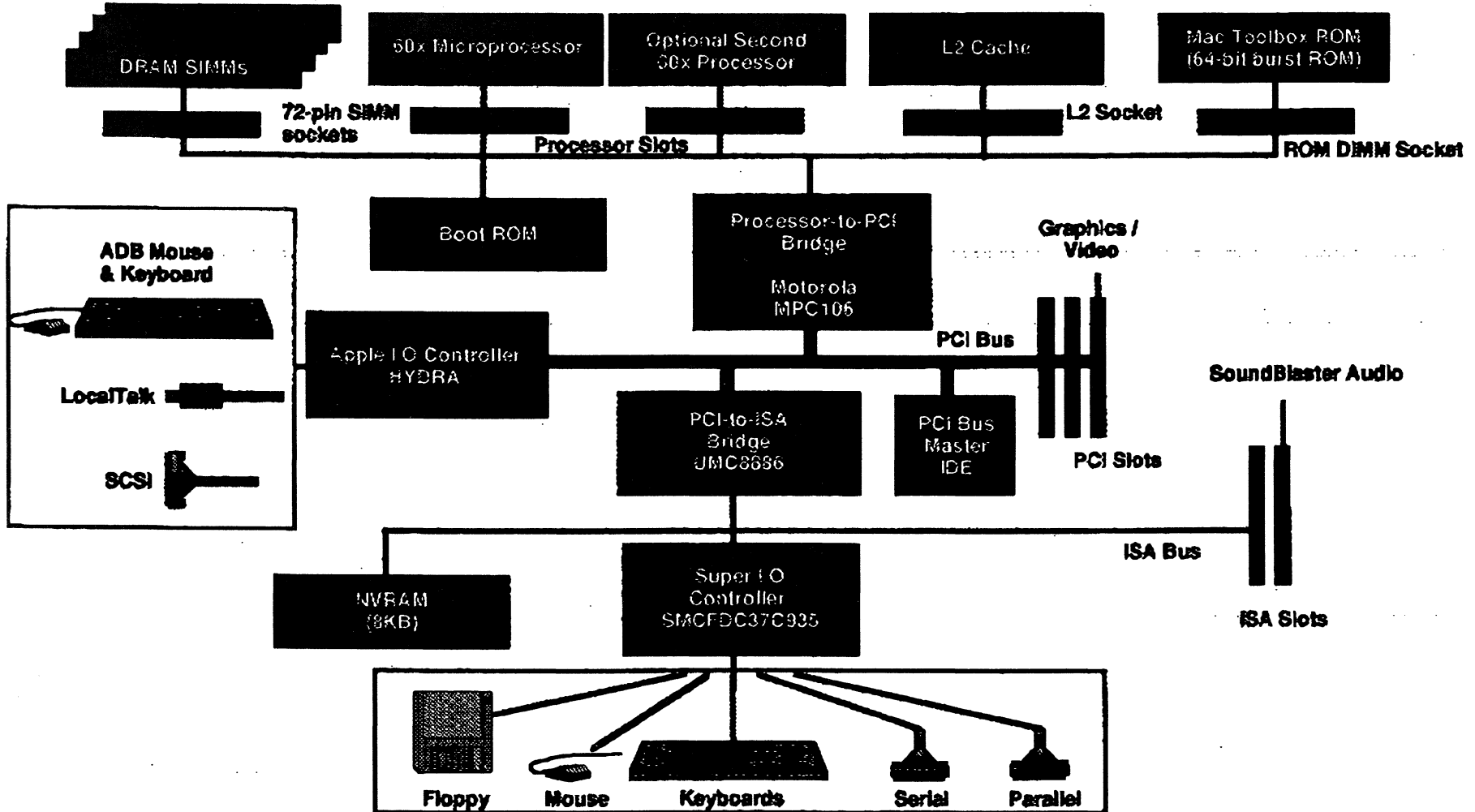
# PowerPC Platform Block Diagram

64



# PowerPC Platform Initial Implementation Example - Yosemite

Available from Motorola SPS



69

**PowerPC™**

PowerPC Platform PRESENTATION - 5

Motorola Confidential Proprietary

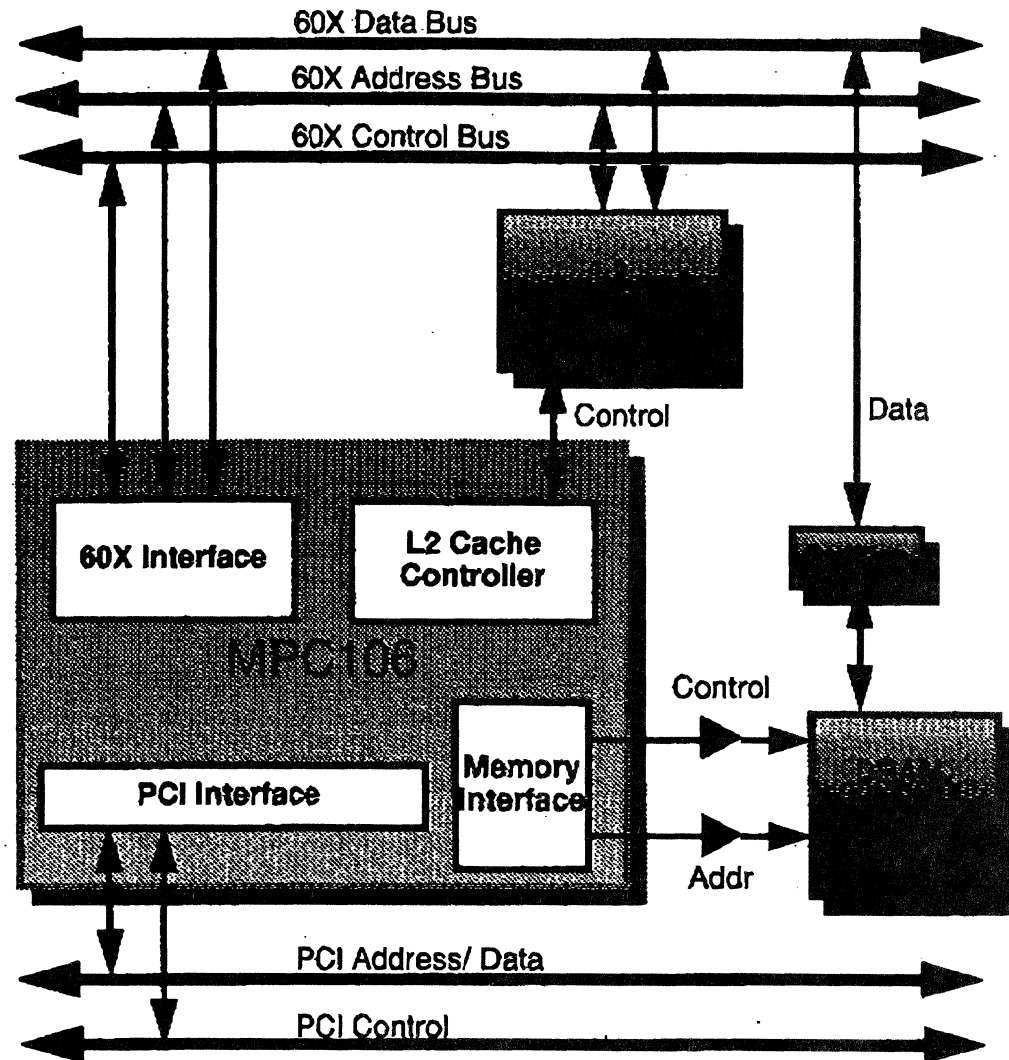


**MOTOROLA**

# Processor to PCI Bridge

## ▼ PowerPC 60X to PCI Bridge

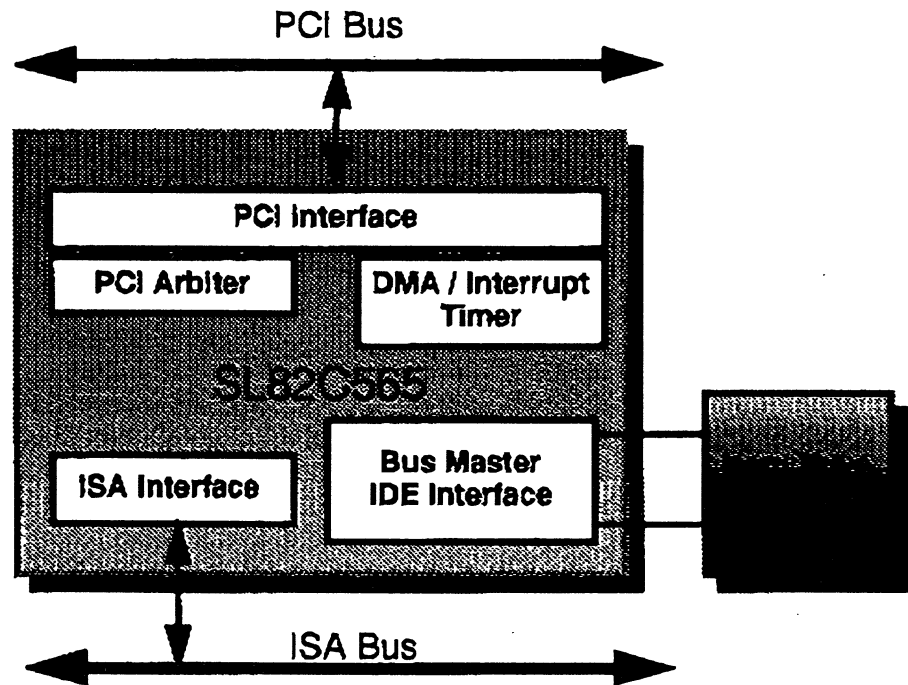
- Available from Motorola
- PowerPC Platform Compliant
- Interface to 601, 603 and 604
- Integrated DRAM controller for EDO and Page Mode DRAM
- Integrated L2 cache controller for Asynchronous and BurstRAM.
- PCI Interface
- Power Management Logic
- Parity or ECC support
- .5µm CMOS, 3.3 Volt
- 304 pin Ball-Grid Array Package
- 1st silicon in August,95
- Mass Production Q1,96



# PCI to ISA Bridge

## ▼ PCI to ISA Bridge

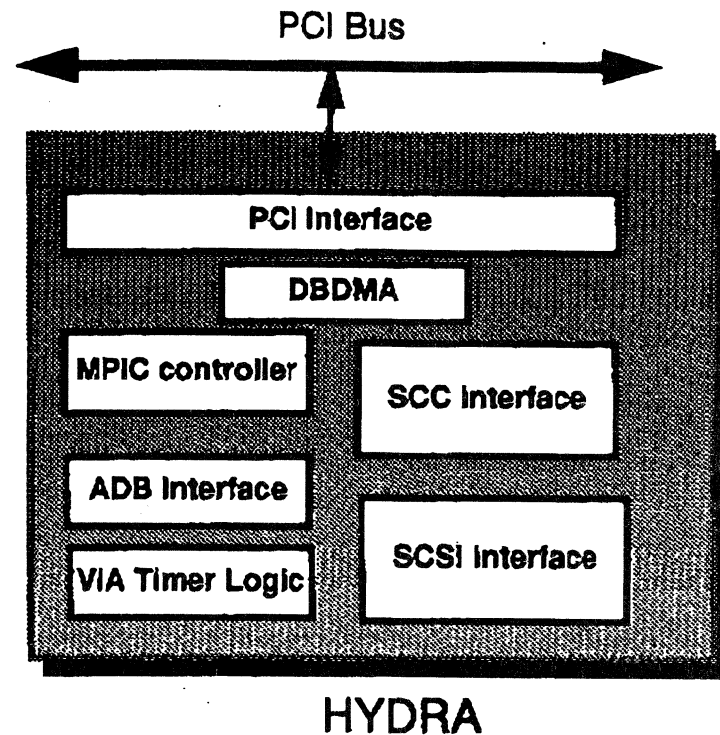
- Available from Winbond Systems Lab
- PowerPC Platform Compliant
- PCI (Rev 2.1) Compliant
- Integrated Dual channel bus master IDE controller
- Integrated DMA, Interrupt controller and timer
- Supports PCI arbiter with host bridge and 5 PCI masters
- Power Management Logic
- 208 pin QFP package
- 1st silicon in October



# Apple Peripheral Chip

## ▼ Apple Peripheral Chip (HYDRA)

- PCI Bus interface
- Integrated MPIC controller to support dual processor
- Integrated SCSI controller
- Integrated SCC controller for Local Talk
- Integrated ADB controller for Keyboard and mouse
- 160 pin QFP package
- 1st silicon in October

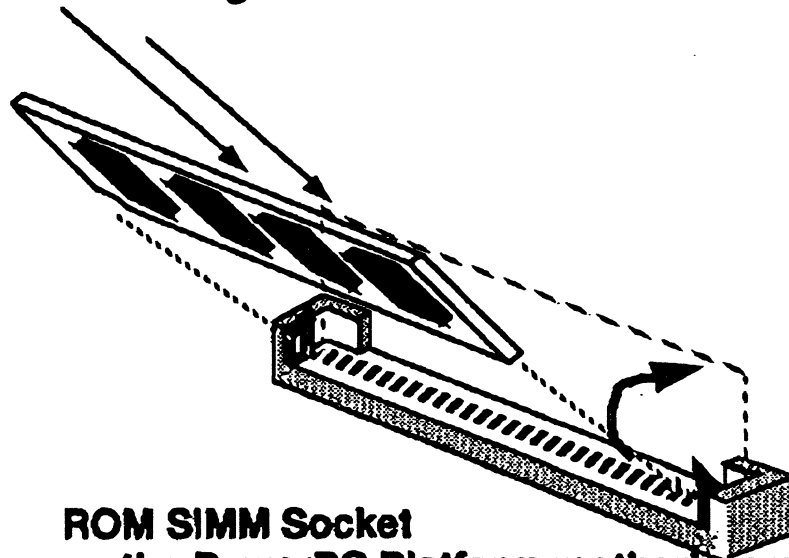


# OS Licensing for PowerPC Platform System

---

- ▼ Apple will license Mac OS, including ROM code, to ANY OEM developing PowerPC Platform system

**ROM SIMM Containing Mac Toolbox**



**ROM SIMM Socket  
on the PowerPC Platform motherboard**

- ▼ Other OS licensing follows normal X86 business model



# Open Firmware Definition

---

## ▼ OS Independent

- Required to Boot Mac OS on PowerPC Platform
- Vener Available to Boot Windows NT
- Bi-Endian Firmware Callbacks

## ▼ Features

- Boot Time Debugger for Firmware and Drivers
- Plug and Play via FCode Drivers

## ▼ Processor/Bus Independent

- Single Peripheral Driver for Multiple Platforms
- IEEE Bindings for PCI and ISA

## ▼ Industry Standard

- IEEE P1275 Standard for Boot Firmware
- ANSI Forth Programming Language Standard

**Digital Semiconductor**  
*The Standard*  
*in*  
**PCI to PCI Bridges**



# 21050 PCI to PCI Bridge

- ❑ **Industries First PCI to PCI Bridge**
  - ✓ Began shipping 1994
  
- ❑ **5.0 Volt 208 Pin Design**
  - ✓ Compliant With the PCI Spec 2.0
  - ✓ Compliant With PCI to PCI Bridge Architecture Spec 1.0
  
- ❑ **Provides 32 Bytes of Buffering in each direction**
  - ✓ Posted Write Data
  - ✓ Prefetched Read Data
  
- ❑ **Supports 6 Devices Without External Logic**
  - ✓ 6 Req/Gnt and Clk Signals

## 21052 PCI to PCI Bridge

- ❑ **Low Cost Variant of 21050**
  - ✓ 3.3 Volt Design - 5.0 Volt I/O Tolerant - 160 Pin PQFP
- ❑ **Supports 4 Devices Without External Logic**
  - ✓ 4 Req/Gnt and Clk Signals
- ❑ **Tighter Clock Skew Specification Eliminates Clock Buffer for Motherboard Applications**
  - No External Buffer or PLL Required
- ❑ **Low Power Dissipation (< 1 Watt) and Low Cost**
- ❑ **Is Compliant With the PCI Spec 2.0 And the PCI to PCI Bridge Architecture Spec 1.0**

# The New 2115x PCI to PCI Bridge Family

- ❑ **Industry'S First PCI to PCI Bridges Compliant With PCI Local Bus Revision 2.1**
- ❑ **Second Generation PCI to PCI Bridge Product Family From Digital Semiconductor**
  - ✓ Digital Semiconductor'S First Generation of PCI to PCI Bridges (21050 and 21052) Are the Accepted Industry Standard Today!
  - ✓ Digital Semiconductor'S Used Its Custom CMOS VLSI Capability and Industry Leading Experience in PCI to PCI Bridge Technology to Develop the 2115X Family

**digital**™

# Industry Leading Features of the 2115x Family

- ❑ Employs PCI Revision 2.1 Delayed Transactions to Provide an Enhanced Buffer Architecture
  - ✓ Enables Buffering of All PCI Transactions
    - Reads and Writes of Configuration, I/O, and Memory Address Spaces
  - ✓ Enables Buffering of Multiple Transactions From Each PCI Interface
  - ✓ Separate Queues for:
    - Posted Write Transactions
    - Read Data
    - Delayed Transaction Requests
  - ✓ Much Deeper Buffers Than First Generation Products

# Other Benefits of the 2115x Enhanced Buffer Architecture

## □ Better Performance Under Load

### ✓ Multiple Buffers Make More Efficient Use of Bus Bandwidth

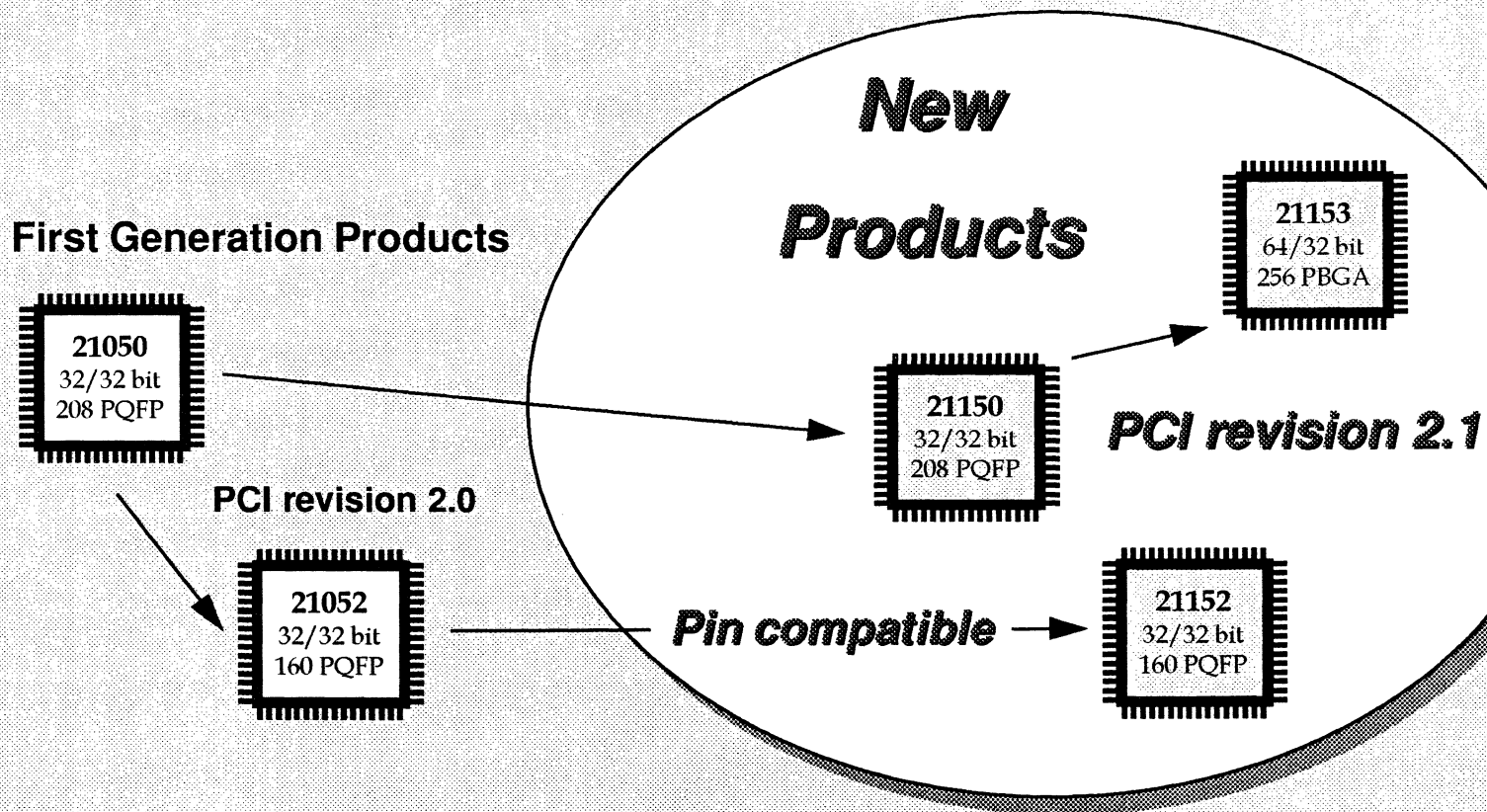
- Less Time Spent Retrying Due to Buffer Full Conditions or Transaction Ordering Requirements

### ✓ Better Traffic/Latency Isolation Between Bus Segments

- Non-Posted Transactions Progress Without Holding the Initiating Bus in Wait States While the Target Bus Is Being Requested

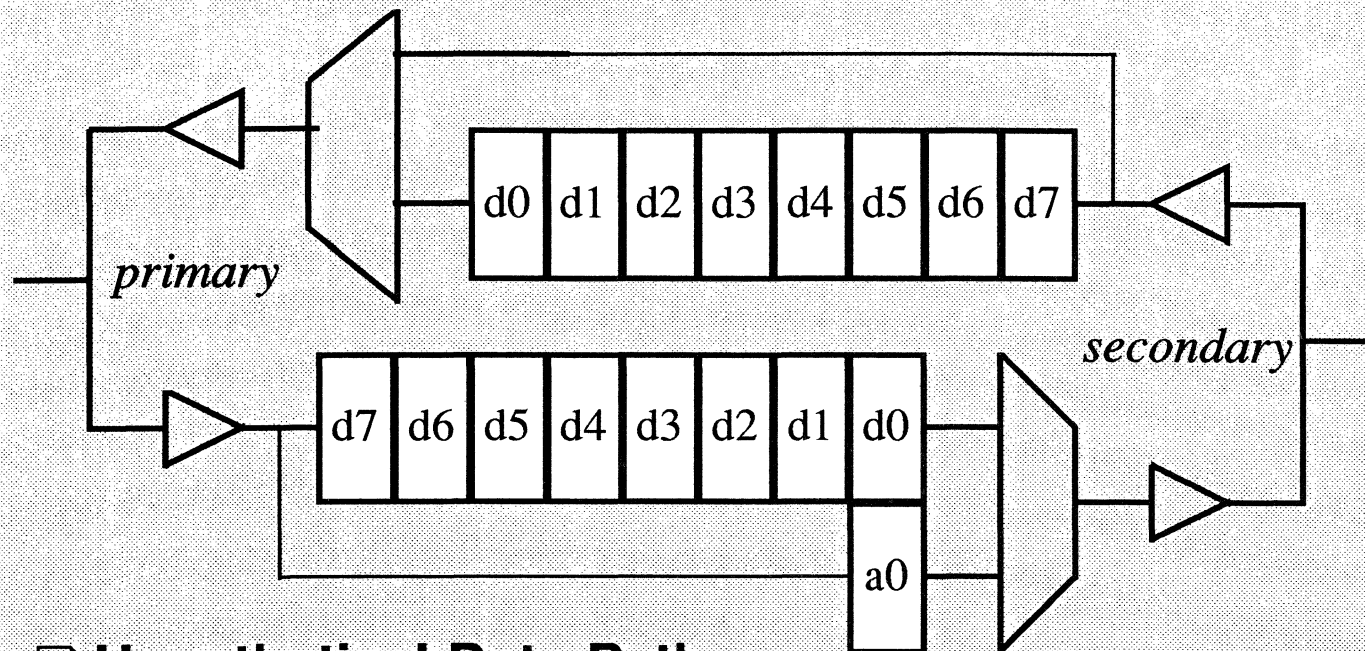
*The Eliminated Wait States Can Be Used by Other Bus Masters on the Initiating Bus Thereby Reducing Latency and Improving Bus Utilization.*

# Digital Semiconductor's PCI to PCI Bridge Product Roadmap





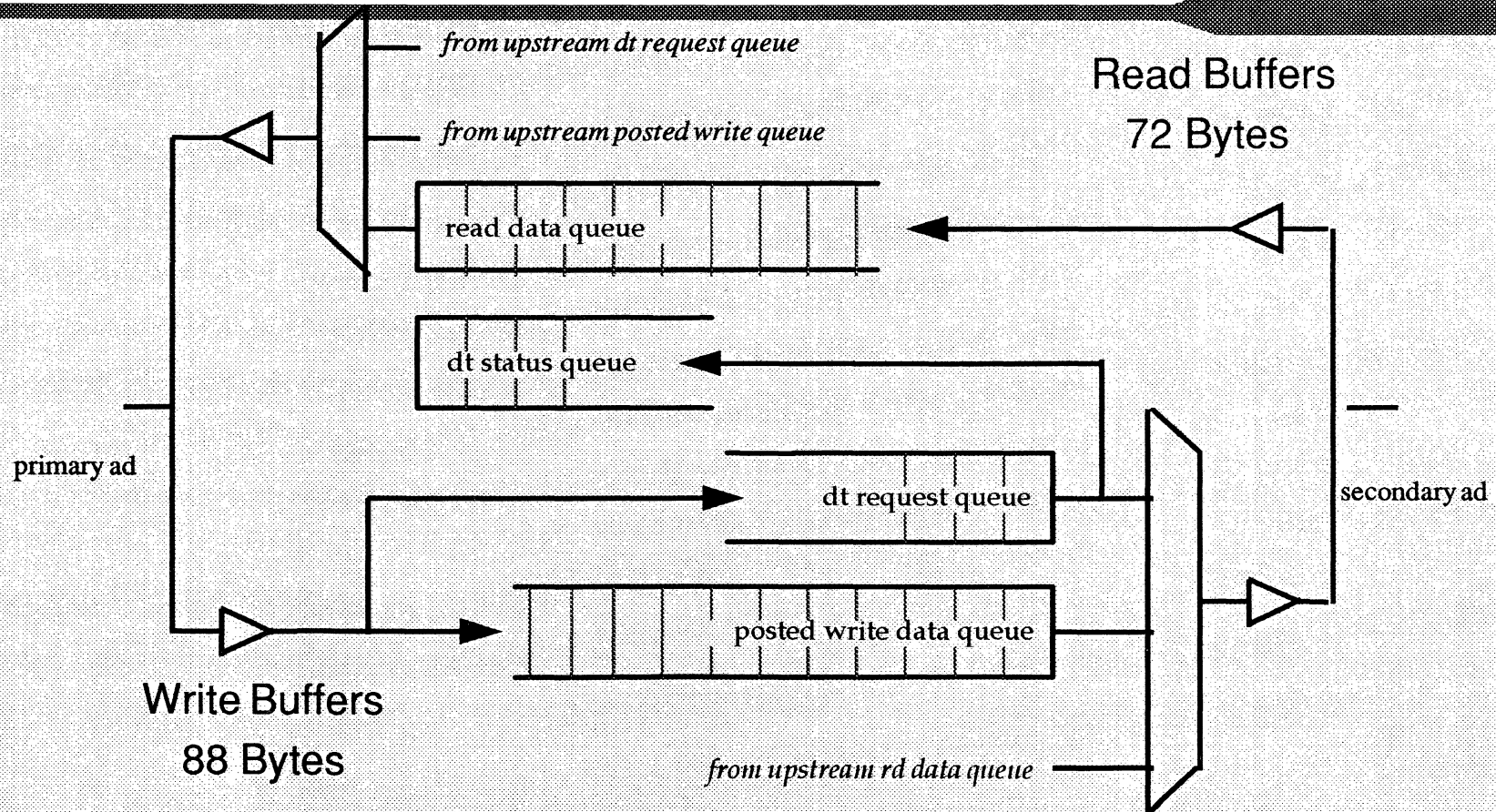
## 21050/21052 Datapath



### □ Hypothetical Data Path

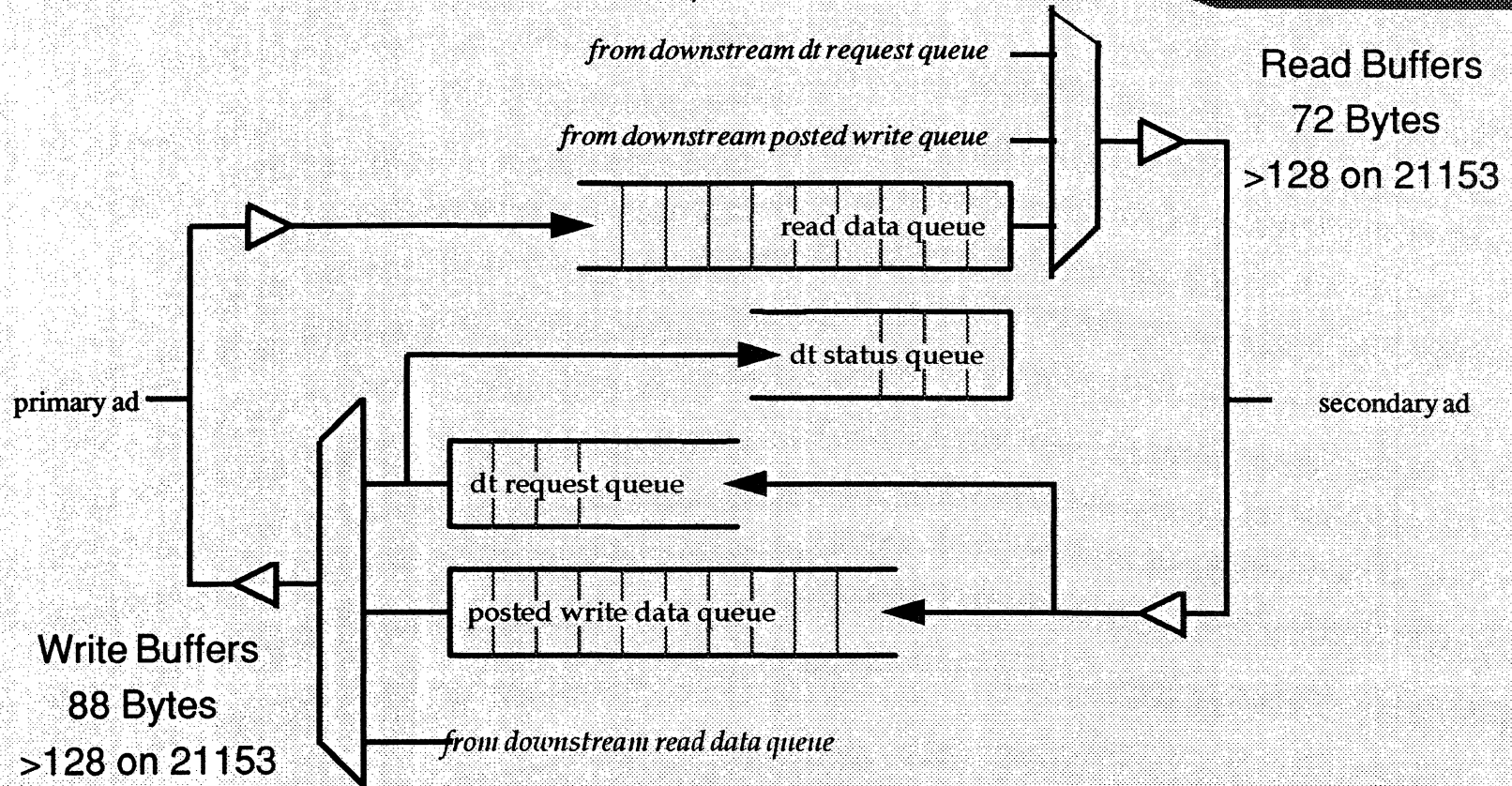
- ✓ 8 Dwords (32 Bytes) of Data Buffers for Each Interface
- ✓ Simultaneous Fill and Drain Can Sustain Burst Transfers at Full PCI Bandwidth (Posted Writes or Prefetched Reads)

# 2115x Family Downstream Datapath



□ Downstream (Primary to Secondary) Data Path

# 2115x Family Upstream Datapath



## □ Upstream (Secondary to Primary) Data Path

# 2115x Family at a Glance

Family Member	Primary PCI Interface Features				Secondary PCI Interface Features					Other Product Features		
	PCI Bus	Write Buffer	Read Buffer	Delayed Transaction Queue	PCI Bus	Write Buffer	Read Buffer	Delayed Transaction Queue	clk, req#, gnt# pins	JTAG	GPIO	Package
21150	32 bit	88 bytes	72 bytes	3 entries	32 bit	88 bytes	72 bytes	3 entries	9 sets	yes	yes	208 PQFP
21152 *	32 bit	88 bytes	72 bytes	3 entries	32 bit	88 bytes	72 bytes	3 entries	4 sets	no	no	160 PQFP
21153	64 bit	88 bytes	72 bytes	3 entries	32 bit	152 bytes	152 bytes	3 entries	9 sets	yes	yes	256 PBGA

\* pin compatible to the 21052

# Common 2115x Family Features

- Programmable Two Level Round Robin Arbitration**
  - ✓ Devices Can Be Individually Assigned High or Low Priority
  
- Low Skew Clock Buffer Outputs**
  - ✓ Enables Direct Clock Connection to PCI Slots
    - Eliminates External Clock Buffers
  
- PCI Universal I/O Buffers**
  - ✓ Allows Connection to 3.3V or 5V PCI Buses
    - Signaling Voltage of Each PCI Interface Is Independent
    - Can Function As Bridge Between 3.3V and 5V PCI Bus Segements
  
- 3.3 Volt Supply for Low Power Consumption**
- 33 Mhz PCI Revision 2.1 Compliant**

# Evaluation Boards

- Evaluation Boards Available for 21150, 21152 and 21153**
  - ✓ PCI Adapter Card With Four Expansion Slots
  - ✓ Two Pals Allow Customer to Experiment With Special Features
  - ✓ Full Documentation and Schematics
  - ✓ Spice Models Are Available
  
- Potential Customers Can Test Designs With Eval Boards Before Beginning Design**

## Design and Application Support

- ❑ **Digital'S PCI Design and Applications Group Provides Customer Support From Evaluation Through End of Life**
- ❑ **Design, Application, Quality/Reliability, Test and Manufacturing Are All in One Location**
  - ✓ Allows Fast Response to Problems

**digital**<sup>™</sup>

## Conclusion

- Digital Is the *Industry Standard* in PCI to PCI Bridges
- Digital Has the Most Expertise in PCI to PCI Bridges
- Digital Has the Products Today and in the Future to Meet Market Requirements
- Digital Can Provide the Application, Test and Design Support Needed to Insure a Successful Product



## DESIGN ISSUES FOR PCI-TO-PCI BRIDGES

Thomas L. Anderson and Mark W. Knecht  
Virtual Chips, Inc.  
2107 North First St.  
San Jose, CA 95131  
(408) 452-1600  
e-mail: toma@vchips.com  
e-mail: markk@vchips.com

Jacques Wong  
Advanced Micro Devices  
P. O. Box 3453, M/S 59  
Sunnyvale, CA 94088  
(408) 749-4918  
e-mail: jacques.wong@amd.com

### **Abstract**

*As use of the PCI Bus has become more widespread, the number of devices needing PCI support and the number of PCI buses per system has increased. The PCI-to-PCI bridge has developed from a niche technology into an essential component in many types of systems. Design of an effective PCI-to-PCI bridge is not straightforward. This paper discusses some of the issues facing bridge designers and some key decisions that must be made to develop an effective solution. The topics discussed include PCI 2.1 requirements, bridge latency, support for asynchronous clock domains, interrupt handling and support for PC "legacy" devices behind a bridge.*

### **Introduction**

The PCI Bus has been widely adopted for a number of excellent reasons. Its precise specification and rich suite of transactions were key factors for early adopters. However, the physical expandability of the bus was limited by the drive characteristics and loading specifications chosen in the original PCI definition. Most manufacturers have found that modern silicon and packaging technologies limit their product solutions to no more than three compliant loads on the PCI bus. While this limit was acceptable for the first PCI-based personal computers, it is insufficient to support many applications demanded by today's system users.

The PCI SIG, working together with some of the interested silicon and systems manufacturers, developed a specification for the first PCI-to-PCI bridge. This specification was released in April of 1994 and has proven valuable in the rise of both commercially and privately developed bridging solutions. Through the use of PCI-to-PCI bridges, system manufacturers have been able to provide the extra PCI connectivity that is needed in high-end server applications and is becoming more necessary in today's desktop solutions.

This flexibility of the PCI specification has proven to be an additional incentive for adoption of PCI. Bridges provide a virtually unlimited ability to add additional PCI buses in hierarchical fashion. However, factors such as the latency of transactions across bridges and the flexibility in clocking the downstream buses can have a major effect on the performance and utility of a multi-bridge system.

The design of a PCI-to-PCI bridge entails tradeoffs in a number of key areas. Failure to make these tradeoffs correctly has resulted in the commercial failure of some bridging based product solutions. Further, the continued evolution of the PCI specification and the increasing demands of PCI-based systems have produced some new challenges for designers. This paper reviews some basic design issues for PCI-to-PCI bridges, discusses some recent changes in the types of bridging solutions

available, and outlines some major issues for the future.

### ***Baseline Features and Issues***

A basic PCI-to-PCI bridge supports two complete PCI buses; the *primary* bus is closest to the host processor and the source of configuration transactions while the *secondary* bus is effectively produced by the bridge itself. The bridge must be capable of acting as a master or target on either bus in a complementary fashion. When the bridge acts as a target on either bus, it must act as a master on the other bus in order to pass transactions to the final target. As such, a pure bridge provides no peripheral device functionality itself but acts merely as an agent to propagate transactions from one PCI bus to the other. When no communication is needed between the primary and secondary buses, a PCI-to-PCI bridge must allow independent, concurrent transactions on both buses. It is only when a device on one bus needs to communicate with a device on the other bus that the bridging function is activated.

Even this baseline level of functionality raises some interesting design issues. The distinction between the two buses means that a bridge generally will not have equivalent interfaces on its two sides. The primary side interface must be capable of handling configuration transactions from the host processor and either taking appropriate action itself or passing the transactions on to devices on the secondary bus. This distinction in the operation of the primary and secondary interfaces of the bridge creates limitations in the use of the bridge.

For example, a PCI-to-PCI bridge is not very useful as a mechanism for connecting two separate, but equal processing environments. It is capable of accepting configuration transactions only from the host connected to the primary bus, relegating the other host to a subordinate status. A more intelligent mechanism for differentiating the upstream and downstream addressing environments would be valuable in some system architectures.

Since the PCI-to-PCI bridge effectively creates the secondary bus, it usually provides some basic support features. For example, the bridge may include the arbitration logic for the secondary

bus so that a dedicated external arbiter is not required. This is entirely analogous to the arbitration support provided by the chipset or host interface that creates the primary PCI bus. The bridge will often provide the clock for the secondary bus if it is to run synchronously with the primary bus. The bridge can buffer the primary side PCI clock and provide multiple copies in order to support the other devices on the secondary bus.

### ***PCI Timing Design***

Of course, the bridge designer must also deal (twice) with the baseline design challenges of any PCI-compliant interface. Operating within the 7ns setup time on key signals can be one of the most daunting tasks in many PCI projects. It is most often the address decoding that consumes the largest portion of the available setup time. Since bridges that attempt to be compliant with all optional aspects of the PCI-to-PCI Bridge Architecture Specification have multiple address ranges to decode, the difficulty of meeting the setup time is greater than for a standard PCI device.

Care should be taken in the specification of a PCI-to-PCI bridge to define its target environment and applications. Eliminating address decoders that are not necessary in the target system will help the designer to meet the PCI setup timing. As with any PCI interface, best performance is achieved with the fastest assertion of the device select and target ready signals.

If, for instance, a bridge is to be used in an environment in which memory addressing is limited to 32 bits, then support of the Prefetchable Upper 32 bit registers (Base & Limit) should be excluded from that bridge's configuration space. A similar trade-off can be made with the I/O addressing when bridging in an X86 based machine. Since the X86 compatible processors provide addressing capability for only 64KB of I/O space, there is no requirement to support the I/O Base and I/O Limit Upper 16 Bit registers. While each of these decoders by itself might appear to be an insignificant burden on the performance design of a bridge, their cumulative effect is often great and can affect both the timing specification and the timing margin in the final product.

The clock insertion delay from the clock input pin to the state elements actually helps meet PCI bus setup requirements. However, the PCI bus also has strict requirements for hold time. It is fairly common to ease the hold time problem by using a minimally-delayed input clock for signals from the PCI bus, and then using clock buffers or a clock trunk to provide clocks to the rest of the bridge. A small clock insertion delay minimizes the skew between the input clock and the clocks at the internal state elements, thereby reducing the chance of races and hold time violations. As with any PCI interface design, delay elements are usually required on the fastest paths. Additionally, the PCI requirement for correct operation at very low frequencies prevents the use of a Phase-Locked Loop (PLL) to reduce internal clock skew

Latency across the bridge is another dimension of performance affected by the requirements of PCI timing. Even if a bridge achieves fast assertion of device select it still can be difficult to achieve a one-cycle latency from one bus to the other. The combination of the PCI bus 7ns setup time and 11ns clock-to-out requirement means that very little time is available around a single register stage to perform all logical functions needed for bridge operation.

#### ***Burst Performance of PCI-to-PCI Bridges***

For PCI systems to obtain maximum performance, the PCI bus must be able to operate as it was intended when originally specified: as a burst bus. Reasonable burst performance is best achieved when the bridge holds multiple data words in FIFOs. Commercially available PCI-to-PCI bridges, to date, have had very shallow FIFOs and this has hindered the burst performance of the bus whenever the bridge is either the target or the initiator of the transaction. Technical papers presented at WinHEC and elsewhere have suggested that, for some chipsets to perform at or near the limits of the PCI specification, burst length should be at least 32 double-words of data. While this suggests that FIFO structures in a bridge should be at least this deep, that is only part of the story.

For more optimum performance, bridge FIFOs should be deep enough to allow for arbitration of the target PCI bus without stalling the device on

the initiating interface. The arbiter for the primary PCI bus, for fairness reasons, provides no special treatment for the bridge. Assuming that any given PCI interface may have as many as four devices operating on it, the bridge may have to wait for all other devices on the interface to access and transfer using the PCI bus prior to gaining access for its own purposes. If system software sets up the PCI device's latency timer values high enough to satisfy the burst performance criteria of the chipset then the bridge may find itself waiting for many tens, if not hundreds, of PCI clock cycles, prior to beginning its transfers on the bus. As I/O bandwidths available from disk and networking interfaces increase, and other PCI devices opt for longer bursts, PCI-to-PCI bridges will be required to significantly deepen their FIFOs to support high bandwidth I/O streams operating across their interfaces.

Having multiple FIFOs is also an essential part of a robust bridge design. It is possible, with a good deal of complex logic, to configure a single FIFO to handle a variety of transactions at the same time. It is simpler and better for performance to have four FIFOs, a separate read and write FIFO for each of the two directions (primary target to secondary master and secondary target to primary master). Separating the FIFOs in this way fosters a clean design style in which the control and datapath for the two directions are symmetrical and as independent as possible.

#### ***Maintaining Data Consistency on Interrupts***

As FIFOs are deepened in PCI-to-PCI bridges to improve burst performance, new problems are seen in the area of interrupt processing. PCI-to-PCI bridges are not required to handle any of the interrupts generated by the devices downstream of them. The bridge user community has identified system problems that were caused by interrupts being delivered to the system prior to the data being delivered to memory.

In systems that use token passing techniques to improve interrupt handling performance, a device may generate a token and write it to memory, and then generate an interrupt to inform the processor that it has completed its task. This sort of operation can significantly improve interrupt handling, especially when

interrupts are shared. If the token is still residing in a posting FIFO internal to the bridge when the interrupt hits the processor, then the processor will check memory and not find the token, at which point the interrupt is effectively lost.

This problem is potentially best solved in the device that generates the interrupt itself, as long as the bridge behind which it resides is properly designed. If the device that has written the token to memory performs a read of the token location in memory, then the bridge is forced, by ordering rules, to flush the FIFO before allowing the read transaction to proceed, thus guaranteeing correct operation. This sort of improvement in the basic system level operation of PCI devices will help, but PCI-to-PCI bridges will continue to operate with 2.0 compliant devices behind them that do not implement these sorts of safeguards. It is for this reason that bridge designers should consider implementation of interrupt handling logic for maintaining the consistency of the data in memory.

New PCI-to-PCI bridge architectures could eliminate some of the problems discussed above through the use of intelligent interrupt monitoring and gating circuitry. This circuitry could be as simple as flushing FIFOs anytime an interrupt occurs. While this may seem like unnecessary overhead to the bridge designers today, the handling of hardware interrupts in systems of the future will certainly be more complicated as the number of PCI devices increases.

### ***Challenges of Asynchronous Design***

Most hostbus-to-PCI and PCI-to-PCI bridges are designed synchronously, with both sides of the bridge running on the primary side clock. This has resulted in some undesirable effects due to the secondary bus being tied to the primary bus. In some cases, as processor speed has increased, I/O performance has decreased if convenient clock multiples were not available. For example, a 100-MHz processor conveniently drives a 30nS PCI bus while a 125-MHz processor conveniently drives a 40nS PCI bus. With synchronous PCI-to-PCI bridge designs, this means that all PCI buses in the hierarchy would be running at 25-MHz, or nearly 25% slower than the I/O devices on these buses were designed to run. While PCI itself was supposed

to decouple the design of I/O devices from the design of the processor, this is one area where their performance may be definitively linked. Because of situations such as these, it is desirable to have the option to support independent clocks on both sides of a bridge.

If the two sides of a PCI-to-PCI bridge are to have completely independent clocks then the designer must pay careful attention to all the issues associated with asynchronous design. This would include not only the data path, but all areas involving signaling between the primary and secondary state machines. The presence of FIFOs in the bridge can aid in the synchronization process; each FIFO may be filled at one clock rate and emptied in the other clock domain. While asynchronous design does pose some difficulties in today's synthesis-centered design environments, it should be addressed if the highest performance metrics are to be met.

It is convenient for the clocking of the PCI devices downstream of a PCI-to-PCI bridge to be handled by the bridge itself. Today's successful bridging products offer a reasonable minimum level of support for synchronous buses. Bridge clocking circuits can be enhanced to include support for externally provided asynchronous clocks and control of clock outputs for power management.

### ***Challenges of PCI 2.1***

The introduction of the 2.1 revision of the PCI Specification introduced a few new challenges to all PCI designers. Specifically for bridges, the delayed transaction feature has proved to be a major issue. This is evident from the delays suffered by the PCI-to-PCI Bridge Working Group as they have worked diligently to release the 2.1 revision of the PCI-to-PCI Bridge Architecture Specification.

Delayed transactions were developed to provide a more bounded time period for individual transactions on the PCI bus. Whereas in the previous revisions of the specification there was no limit to the amount of time that a target might hold the bus prior to beginning the transfer of data, the 2.1 revision of the specification placed a 16 clock limit on the delay until the first data transaction. If a device finds itself unable to

begin transfer of the data in that time period, then the device must store sufficient information to allow it to release the PCI bus, continue on with the transaction, recognize the same transaction when it is retried and then respond appropriately. A read or write transaction requires that the address, command and byte enables be stored, while a write also requires the data to be saved to insure proper response to a retry.

This mechanism provides individual PCI devices with more potential access to the bus, with no additional silicon overhead, if they can always guarantee transfer of at least the first data transaction within this 16 clock boundary. For bridges though, the story is quite different. The very nature of a bridge means that it has no data on the devices operating behind it, and in fact may be posed with many layers of PCI bus hierarchy. The bridge therefore cannot guarantee a response within the 16 clock limit and must have significant delayed transaction capabilities if it is to maintain system performance. This aspect of operation has significantly complicated bridge design in the short term.

Delayed transactions in a PCI-to-PCI bridge fall under a number of categories as they proceed toward completion. The PCI specification defines five such categories:

- PMW, or Posted Memory Write
- DRR, or Delayed Read Request
- DWR, or Delayed Write Request
- DRC, or Delayed Read Completion
- DWC, or Delayed Write Completion

The PCI specification determines that any 2.1 compliant device *may* create a queue for these delayed transactions, and that the queue *may* be as deep as required. The specification then further defines that some transactions *may*, for performance reasons, be allowed to complete out of order, at least from the point of view of the bridge itself. This mechanism is known as *The Ordering Rules*.

The operation of a specific bridge product defines which of the optional Ordering Rules the bridge will implement. Extreme care must be taken in this area of the specification since potential deadlock or live lock conditions can

arise. Significant work was done by the PCI SIG for the release of the 2.1 specification, and still, it is reported, that the PCI-to-PCI bridge Working Group found many inconsistencies in these rules with respect to PCI-to-PCI bridges. The bridge designer must be wary of the problems posed, work through all possible scenarios, and test these cases as thoroughly as possible in simulation.

### ***Legacy Devices Behind Bridges***

As the PCI bus has become nearly ubiquitous in PC systems, many designers are anticipating the gradual phase-out of the ISA bus. However, it seems unlikely that the ISA bus legacy devices (serial port, parallel port, floppy disk drive, interrupt controller, etc.) will disappear even as the ISA bus itself becomes extinct. The standards for these devices date back to AT-class systems and users have come to rely on their presence and compatibility. Since these devices have specific hard-wired addresses, they cannot be simply mapped to new locations by PCI configuration commands, nor can they be moved to new addresses without sacrificing AT compatibility. The original Bridge Architecture Specification takes this into account by specifying that none of the legacy addresses required for AT compatibility should exist downstream of a PCI-to-PCI bridge unless the bridge is configured for ISA compatibility. In this model all ISA compatibility must exist in one, and only one, portion of the system bus hierarchy.

A new set of specifications, developed by a group of silicon and system companies, proposes the use of two evolutionary technologies. These technologies are known as *Distributed DMA* and *Serial Interrupts*. With the successful introduction of these proposals, AT class compatibility in the areas of the traditional DMA and interrupt servicing can be maintained even while the peripheral devices are located downstream of a PCI-to-PCI bridge. Unfortunately these two specifications do not tackle the problems associated with truly distributing the ISA legacy devices themselves to distinct locations in the hierarchy. For improved user configurability, resulting in fewer technical support questions, all PCI slots in a machine should meet a minimum level of functionality

independent of the board level product that is plugged into them.

For PC-AT compatible machines, a limited set of I/O address range decoding would need to be supported to allow legacy devices to be moved to any level of the PCI bus hierarchy. While this list may not be extensive, and since some of these technologies may be obsolete in the near future, the list should be used only as a potential checklist. The hexadecimal I/O addresses for the most standard legacy peripherals and system devices are:

- DMA : 000-00F, 0C0-0DF
- IRQ : 020 - 021, 0A0 - 0A1
- Timers : 040 - 043, 048 - 04B
- System Ports : 061, 092
- IDE1 : 1F0 - 1F7, 3F6
- IDE2 : 170 - 177, 376
- Floppy : 3F0 - 3F5, 3F7
- Parallel Ports :  
3BC - 3BE, 378 - 37A, 278 - 27A
- Serial Ports :  
3F8 - 3FF, 2F8 - 2FF

The additional decoding logic required for the legacy devices does imply additional difficulties when attempting to operate within the 7ns setup time requirements of the PCI specification. In most cases the bridge will not achieve fast assertion of the device select signal.

#### ***Arbitration for High Performance Operation***

As isochronous data streams become more commonplace in the PC of tomorrow, devices will need access to the bus in the most timely manner possible. While the delayed transaction capabilities of the 2.1 specification go a long way towards guaranteeing appropriate use of the bus once a device has gained access, this alone will not get a device onto the bus to begin the transfer. A well designed and integrated arbitration unit is necessary if the bridge is to maintain high system performance at the lowest overall system cost. The PCI 2.1 specification outlines a reasonable set of criteria for designing a multi-level arbiter. Future bridge designs should implement this as a baseline set of functionality, and then develop creative ways to help devices gain access based on time-slicing. It would be appropriate for the next revision of the PCI-to-PCI bridge specification to make

arbitration programming a defined mechanism so that system BIOS manufacturers could work toward supporting one solution. Today's bridges use individual programming mechanisms which require specific BIOS or driver support. It would be valuable to have a consistent programming model, supported by all BIOS manufacturers, that could guarantee high performance on the secondary bus of a bridge, without the system integrator having to resort to independent solutions.

#### ***A Checklist for Advanced Operation***

Determining whether a bridge is appropriate for any given system design is of course the responsibility of the system architecture and integration team. The authors offer the following list of points for the team to consider when choosing a PCI-to-PCI bridge device.

- Baseline operation as defined by the PCI SIG is appropriate for the system under development. No additional features are required for correct system operation.
- The bridge is compliant with the memory and I/O addressing portions of the PCI SIG Bridge Architecture Specification appropriate for the system under development.
- The bridge contains data buffering sufficient to support the burst performance of all I/O and computing devices that will be transferring data across its level of the hierarchy.
- The bridge offers appropriate mechanisms for maintaining data consistency during all forms of interrupt processing, be they register based or token based.
- The bridge is able to operate asynchronously across its interfaces to enable the highest performance operation possible on both interfaces.
- The bridge implements an appropriate PCI 2.1 delayed transaction queue for both the upstream and downstream sides of the bridge.

- The bridge allows the system architects an appropriate degree of freedom in the placement of AT class legacy devices, allowing for end user configuration options.
- The bridge integrates all downstream clocking and arbitration functions necessary to support high performance transaction on its secondary interface. The arbitration unit must support the needs of isochronous data streams crossing its level of the busing hierarchy.

PCI-to-PCI bridge designers should have a strong understanding of the system requirements for their bridge products if maximum performance is to be achieved. Bearing in mind that some of the above points may be mutually exclusive, the designers should include those features satisfying those requirements.

#### ***The Authors***

Thomas L. Anderson is Vice President of Engineering at Virtual Chips, the leading supplier of synthesizable PCI cores and PCI verification environments. He was previously an Engineering Manager in the I/O and Network Division of Advanced Micro Devices.

Mark W. Knecht is a senior architect at Virtual Chips, designing PC system solutions. He was previously a Senior Member of Technical Staff at Advanced Micro Devices, where he developed PCI-based multi-function I/O devices.

Jacques Wong is a Senior Designer in the I/O and Network Division of Advanced Micro Devices. He has been a lead engineer on numerous projects involving SCSI, Ethernet and PCI bridge technologies.

# PCI Interrupt Controller for Industry Standard PCI-ISA Bus Architecture using PCI-to-PCI Bridge Technology

**Ross L. Armstrong**

Digital Equipment Corporation (Scotland) Ltd.,

Moseshill Industrial Estate,

Ayr, Scotland. KA6 6BE.

Fax: [44] 1292 883241

e-mail: rarmstrong@nesbit.onot.dec.com

*The demand for more Peripheral Computer Interconnect [PCI] device configurations beyond the limit set in the PCI local bus specification has prompted the development of several PCI-PCI bridge solutions. This paper describes a new PCI Industrial Computer Manufacturers Group [PICMG] PCI-ISA bus architecture implementation using Digital Equipment Corporation [Digital] PCI-PCI bridge technology. Layered PCI bus architectures, PCI interrupt latency implications and performance optimisations for PCI-PCI bridge designs are discussed. Reference will be made to Digital's family of 64 bit Alpha Single Board Computers [SBC] and PCI-ISA backplanes which have been specifically designed to address multiple, low cost, high performance PCI requirements associated with high speed communications and graphics in embedded applications.*

## Overview

Digital's Embedded and Real-time line of business has developed a series of modular computing products supporting an open systems environment based upon the PICMG PCI-ISA SBC standard. Two goals of the Digital Modular Computing Component [DMCC] program were to

- develop a number of PCI based backplane products that would enable customers to procure and configure industry standard PCI and ISA I/O option cards.
- provide extensive PCI I/O option card slots and maintain optimum bandwidth/performance for bridged slots.

Whilst the former requirement was a simple undertaking, the latter provided a greater challenge to the platform designers. Reduction in bandwidth, however moderate, could be encountered due to software or hardware inefficiencies i.e. degraded interrupt servicing (latency) or propagation/timing delay due to the bridge implementation.

The choice of Digital PCI-PCI bridge chip adequately meets the required timing specification, however interrupt lines derived from secondary bus devices are not routed through the Digital PCI-PCI bridge chip.

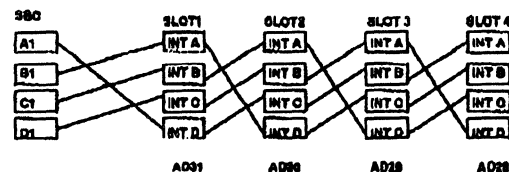
This allowed the designers to thoroughly review and improve upon the standard interrupt binding strategy for PCI buses, where multiple devices might have to share interrupt lines.

The PICMG single board computer connector has only four interrupt lines assigned to it: INTA#, INTB#, INTC# and INTD#, as does each PCI slot connector.

A routing or binding strategy is required to connect between the PCI option I/O and the SBC INTx# line it uses when requesting an interrupt.

In hardware terms, Figure 1 PICMG Single PCI Bus Interrupt Binding, shows how this structure is intended to be provided.

Figure 1 PICMG Single PCI Bus Interrupt Binding



The IDSEL line per slot is assigned to AD[31:28] as per the PICMG Specification. These are used to identify device numbers as given in the configuration address.

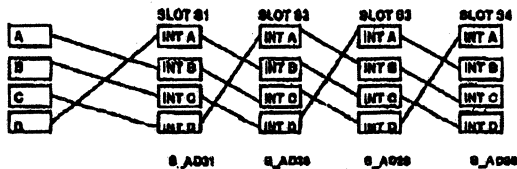


The system firmware (or BIOS) code must assume an interrupt binding architecture for its environment. The PICMG specified binding for the primary PCI bus (PCI Bus 0) is shown in Figure 1, i.e. it is hard coded. Because only the firmware (or BIOS) knows how the PCI INTx# lines are routed to the system controller, a mechanism is required to inform the operating system device driver of an interrupt occurrence. This mechanism typically requires a chained 'software' search of each device using a specific hardware interrupt to identify the source.

This can be achieved by either having the firmware/BIOS poll all PCI devices to determine which originated the interrupt request and then initiate the correct interrupt service routine or alternatively have the operating system kernel interrupt dispatch routine sequentially call each individual device interrupt service routine until the correct source has been identified and serviced. [The latter example is implemented by Microsoft Windows NT].

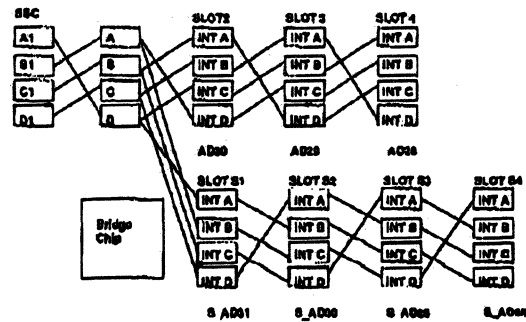
The binding structure becomes even more congested when additional (bridged) PCI buses are implemented in accordance with the *PCI-to-PCI Bridge Architecture Specification Revision 1.0*. Their PCI bus interrupts must be connected as per Figure 2 Secondary PCI Bus Interrupt Binding.

Figure 2 Secondary PCI Bus Interrupt Binding



This secondary binding architecture must be overlaid upon the respective primary slot binding that the bridge now occupies. The net effect being illustrated in the example for one PCI-to-PCI bridge in Figure 3 PCI-to-PCI Bridge Implementation.

Figure 3 PCI-to-PCI Bridge Implementation



The use of the wire-OR [sometimes also known as *shared*] binding design, means that the polling and decode of an interrupt request can be significantly *less than optimal*. The latency for this operation is also *unpredictable* i.e. with N PCI slots, determining the originator of the PCI interrupt request could take a minimum of 1, up to a maximum of (N-1) bus read cycles.

An alternative solution was investigated in order to improve upon this industry standard binding architecture if PCI interrupt latency performance was not to be compromised in large PCI systems. [Any proposal would take cognisance of, and retain support for, the traditional wire-OR scheme.]

The design goal was to provide improved performance while maintaining an open system architecture capable of supporting both existing and alternative modes.

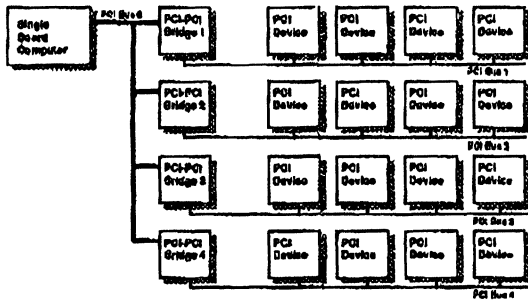
## Interrupt Controller

### Assumptions/Limitations

The interrupt controller must be able to support up to 4 primary PCI devices. A primary PCI device being either a PCI bridge or a physical connector. Each PCI bridge can have up to 4 secondary devices implemented behind the bridge.

The largest configuration would mean a maximum of 16 individual PCI connectors, as demonstrated in Figure 4 Maximum Allowable PCI Configuration.

Figure 4 Maximum Allowable PCI Configuration



This implies a maximum of 64 PCI interrupt sources. A controller that can service all of these product scenarios, or some subset thereof, must do the following:

- support up to 4 primary devices
- be able to identify whether a primary device is either
  - an on-board PCI-PCI bridge (with up to 4 'bridged' secondary connectors behind it)
  - OR
  - a physical connector
- be able to uniquely identify each of the 16 potential interrupts that can be generated from a PCI bridged device (i.e. four interrupts from each of the 4 secondary devices) AND
- be able to uniquely identify each of the 4 potential interrupts that come from a physical connector

This implies that the controller will have the following:

- a register (or similar) to detail whether a primary device is a physical connector or an on-board PCI-PCI bridge
- a register (or similar) for each primary device (i.e. 4 in total) to provide status for each PCI interrupt supported by that primary device. Note that the requirements for a bridged device are very different from a non-bridged device and the format of the register will be different for each case.
- a register (or similar) to identify which primary device caused an interrupt (i.e. to prevent having to read all 4 interrupt registers to determine the interrupt source).

The backplanes developed as part of the DMCC program are intended for use with many operating systems and non-Alpha single board computers.

Copyright Digital Equipment Corporation, 1996

Therefore, they must also be compliant with the shared interrupt scheme as defined in the PICMG PCI-ISA Card Edge Connector Proposal for Single Board Computer [SBC] Specification, Revision 2.0 and PCI-PCI Bridge Board Edge Connector for Single Board Computer Specification.

To meet this two fold requirement the proposed controller supports two unique modes of operation with some means of switching between them. For convenience this was determined to be software selectable.

The default mode, at power-up, makes the backplane compliant to the PICMG specification. This will be known as *PICMG Mode*.

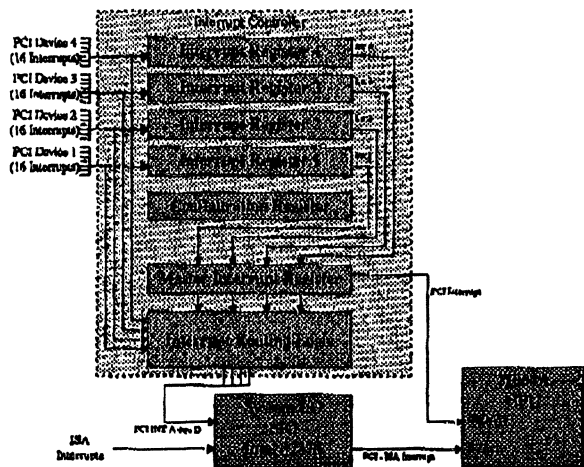
Operating systems [OS] wishing to make use of the interrupt controller must explicitly switch, via software, to the desired mode of operation.

A bonus of this design is that the hardware is (in simple terms) a form of hardware interrupt accelerator usable by multiple operating systems and hardware platforms, if their corresponding BIOS or firmware code is appropriately configured. This mode is known as the *Accelerator Mode*.

#### Generic Architecture

The basic form of the Interrupt controller is shown in Figure 5 DMCC Interrupt Controller Block Diagram. The interrupt controller is split into multiple functional blocks, each section's usage being dependent on the desired mode of operation; either PICMG Mode or Accelerator Mode. These modes are mutually exclusive and are discussed in the following sections.

Figure 5 DMCC Interrupt Controller Block Diagram



The example and illustrations used throughout this paper refer to a specific DECchip 21064A PICMG

PCI-ISA single board computer implementation. The concepts are generic and can be fully utilised by alternative platforms. In Figure 5, the interrupt controller functionality is shown within the shaded area and is physically located on the backplane; the System I/O and CPU being resident on the actual SBC.

### PICMG Mode

This is the default mode for the Interrupt Controller on power-up. The Register Logic blocks are disabled and all inputs are fed into the Interrupt Routing Logic block. It implements the necessary binding to be compliant with the appropriate PICMG specification (as per figures 1 & 2) and addresses the routing for 64 individual interrupt request lines to 4 outputs i.e. the four SBC INTx#. When in this mode the Interrupt Controller appears as shown in Figure 6 DMCC Interrupt Controller Block Diagram - PICMG Mode.

#### Interrupt Registers

The Master Interrupt Register and Interrupt Registers 1 through 4 are not available and have no meaning when accessed (i.e. writes are not stored and reads give indeterminable results).

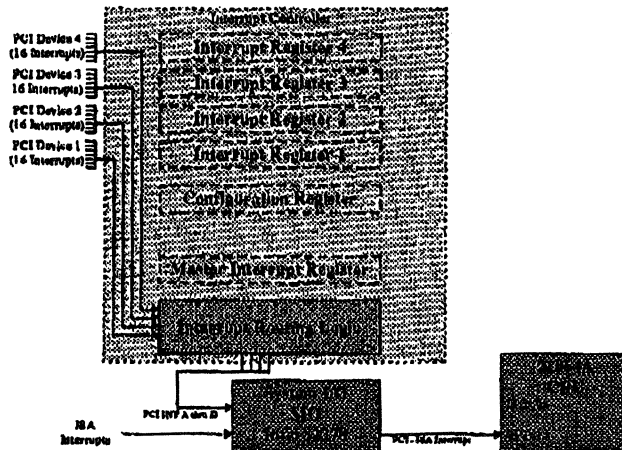
#### Configuration Register

The Configuration Register again has no real meaning in this mode, however it is always active since it is the means to switch to Accelerator mode. See later for details on how this is achieved.

#### PCI Interrupt Routing

In PICMG mode, PCI device interrupts are taken directly to the Interrupt Routing Logic where they are simply wire-or'd, to provide INTA, INTB, INTC and INTD, as specified in the *PICMG PCI-ISA Card Edge Connector Proposal for Single Board Computer [SBC] Specification, Revision 2.0* and *PICMG PCI-PCI Bridge Board Edge Connector Proposal for Single Board Computer [SBC], Revision 1*. These are routed to the System I/O IRQ lines in the demonstration example provided.

Figure 6 DMCC Interrupt Controller Block Diagram - PICMG Mode

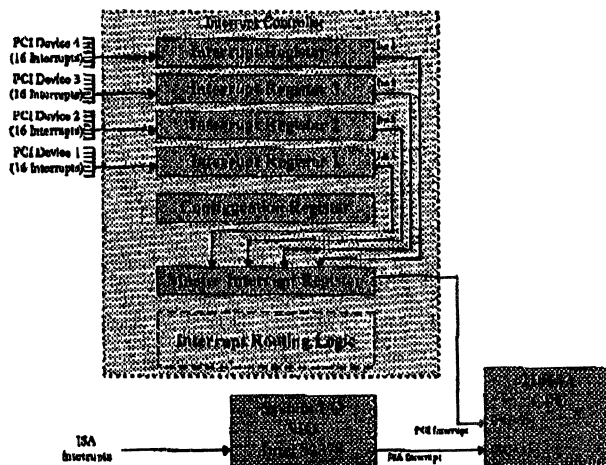


### Accelerator Mode

The interrupt controller or Accelerator Mode can only be enabled via software. To enable this mode, the Control Register must be written to, prior to enabling interrupts.

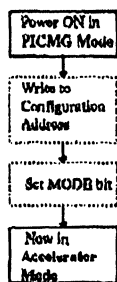
When in this mode the Interrupt Controller looks as shown in Figure 7 DMCC Interrupt Controller Block Diagram - Accelerator Mode.

Figure 7 DMCC Interrupt Controller Block Diagram - Accelerator Mode



The software sequence for enabling Accelerator Mode is shown in Figure 8 Accelerator Mode - Software Enabling Sequence.

Figure 8 Accelerator Mode - Software Enabling Sequence



All Registers are implemented as 32 bit registers addressable in ISA space. [The interrupt controller could as easily have been implemented as a PCI device, however it would then be counted as a full PCI device load and could have had an adverse impact on the total 10-load limit].

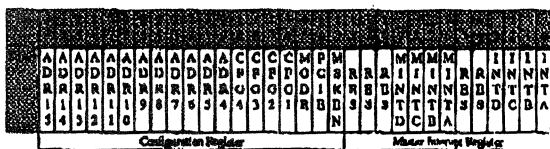
### Configuration and Master Interrupt Register

Table 1 Configuration and Master Interrupt Register defines the register bit allocation. The Configuration Register is always active and is the only means of controlling the Interrupt Controller's behaviour. The Configuration and Master Interrupt Register is located at ISA I/O address 0500h - 0503h.

Apart from having the mode enable bit [MODE], it also stores the high order ISA I/O address bits for Interrupt Registers 1 through 4 [ADR[15:4]]. The low order address bits [ADR[3:0]] are fixed at 0000, 0100, 1000, 1100 respectively.

The backplane configuration details are stored in CFG[4:1], (bits [19:16] of the Configuration Register), defining which primary PCI slots are connectors and which are bridge chips i.e. which have four versus sixteen potentially active interrupt lines.

Table 1 Configuration and Master Interrupt Register



In PICMG Mode, the PCIE bit defines whether the four INTx# interrupts are routed to the System I/O or whether the one PCI interrupt line is routed directly to the CPU. In typical PICMG applications ISA interrupts are heavily used and the PCIE bit can free up to four ISA interrupts. It is always set in Accelerator Mode.

MSKEN is used to support interrupt polling. When enabled the interrupt status bits in the four interrupt registers are dependent upon their corresponding

MASK bits. When disabled they match the status of the interrupt source.

The Master Interrupt Register is only enabled when in accelerator mode. This register gets its input from the 4 Interrupt Registers and is used to determine which Interrupt Register should be read to find the source of the PCI interrupt.

INT[D:A] reflects the status of the corresponding Interrupt Register[4:1] i.e. INTD status is the logical OR'ing of the sixteen Interrupt Status bits stored in Interrupt Register 4. INT[D:A] can be correspondingly masked by MINT[D:A].

In this way the PCI interrupt source can be determined in two ISA read cycles; one to the Master Interrupt Register and one to the specific Interrupt Register.

### Interrupt Registers[4:1]

Each of the 4 Interrupt Registers represent a primary PCI device. The following table maps the primary PCI device to it's associated Interrupt registers. The address of those Interrupt registers is defined by the contents of the ADR bits in the Configuration Register.

Table 2 Interrupt Register Mapping defines the Configuration Space Address for each of the primary PCI devices and gives an example of possible ISA I/O address' for each Interrupt register.

Table 2 Interrupt Register Mapping

Primary PCI Device	Associated Interrupt Register	Primary PCI Device Config Space Address	Interrupt Register ISA I/O Address
1	1	AD31	0510h - 0513h
2	2	AD30	0514h - 0517h
3	3	AD29	0518h - 051Bh
4	4	AD28	051Ch - 051Fh

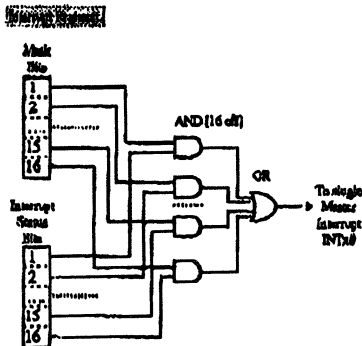
The exact format of each interrupt register is dependent on whether a primary PCI device is a physical PCI connector or a PCI-PCI bridge. The format of Interrupt Register 1 through 4 is defined by the CFG bits within the Configuration Register. This can be used to determine the exact configuration of the backplane, and hence the number of potentially active interrupt lines. Table 3 Interrupt Registers[4:1], defines the register bit allocation.

Table 3 Interrupt Registers[4:1]



If the primary PCI device is a connector the Interrupt Register only requires to store the status and MASK bits for four interrupt lines i.e. only bits [3:0] and [19:16] have any meaning.

Figure 9 Interrupt / Mask Operation



When the primary PCI device is a PCI-PCI bridge, the corresponding Interrupt Register must store the status of up to 16 interrupt lines for 4 secondary connectors implemented behind the PCI-PCI bridge and also the MASK bits for each individual interrupt line i.e. all [31:0] bits are valid.

The interrupt STATUS bits [15:0] are AND'd with their corresponding 16 Interrupt Register MASK bits. The results of each AND operation are then OR'd together to form a single INTx# signal that is routed to the Master Interrupt Register, as illustrated in Figure 9 Interrupt / Mask Operation.

Note : If a multi-function option card (i.e. an option with a bridge) is plugged into a physical connector, there is support for the 4 primary interrupts from behind it's on-board bridge. If more than four interrupts are used (i.e. via sharing) they are not supported.

## Accelerator Interrupt Decode

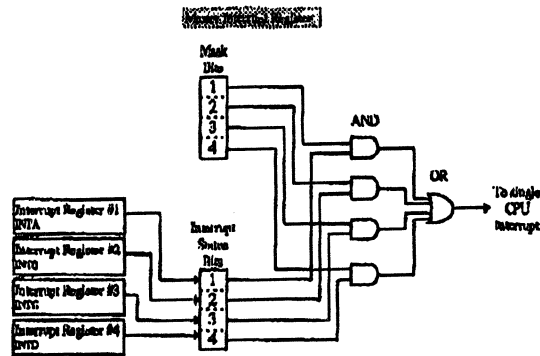
### Hardware Interrupt Architecture

In Accelerator Mode, INT[D:A] in the Master Interrupt Register reflects the status of the corresponding Interrupt Register[4:1] i.e. INT[D:A] status is the logical OR'ing of the sixteen Interrupt Status bits stored in each Interrupt Register [4:1].

This two stage interrupt register strategy allows rapid decoding of the interrupt source without expanding any individual register set beyond 32 bits.

Copyright Digital Equipment Corporation, 1996

Figure 10 Interrupt Decode Schematics



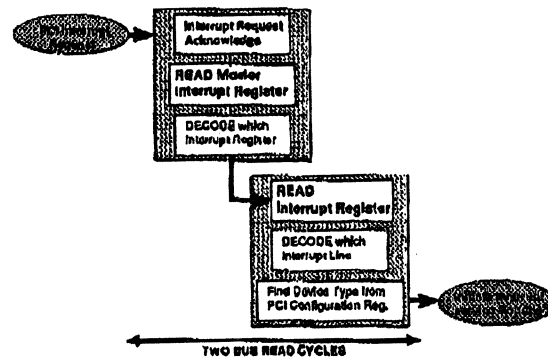
The INT[D:A] interrupt status bits are AND'd with their corresponding 4 Master Interrupt Register MASK bits. The results of each AND operation are then OR'd together to form the PCI interrupt request signal that is routed to a single IRQ on the CPU, as illustrated in Figure 10 Interrupt Decode Schematics.

The final logical routing of the Master Interrupt register is not limited by the MSKEN bit status. Routing of the INT[D:A] interrupts is only determined by the value of the interrupt status bit and it's corresponding mask bit.

### Firmware/BIOS Interrupt Decode

The interrupt accelerator decode architecture influences the host CPU firmware/BIOS, and is usually transparent to the target operating system. The firmware/BIOS must implement a decode routine as per Figure 11 Software Decode Steps.

Figure 11 Software Decode Steps - Accelerator Mode



The predictable and repeatable time to dispatch the appropriate interrupt vector (service routine), after receipt of an interrupt request is two bus read cycles. The decode operation logically occurs in parallel with the read cycle.

## Interrupt Latency

The interrupt *dispatch latency* is the elapsed time from receipt of an interrupt request to dispatch to the interrupt service routine.

The interrupt *service latency* is the elapsed time from entry of the interrupt service routine to its completion.

Exact interrupt latency, for a given system configuration, will be operating system dependent i.e. the interrupt *service latency* may vary significantly between operating systems even when the *dispatch latency* in firmware/BIOS is identical.

Operating systems vary in interrupt service routine efficiency and can be equally dissimilar across hardware platforms. The interrupt accelerator optimises the hardware aspect of this process.

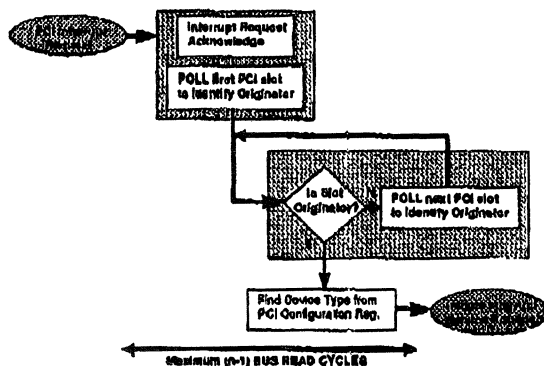
### PICMG Mode

The wire-OR'd binding strategy is not optimal in large PCI slot configurations and most PCI-PCI bridge implementations.

It directly impacts the achievable *interrupt dispatch latency*, and some interrupt service methodologies [e.g. round robin, etc.] can further reduce efficiency in these types of environments.

The dispatch latency is also *unpredictable* i.e. with  $N$  PCI slots, determining the originator of the PCI interrupt request could take a minimum of 1, up to a maximum of  $(N-1)$  bus read cycles.

Figure 12 Software Decode Steps - PICMG Mode



The interrupt *dispatch latency* in very large systems can result in severe degradation of the system performance. This is caused by I/O devices 'stalling' because they cannot get serviced efficiently. In extreme configurations, a particular device may 'never' get its interrupt serviced, resulting in failure of that

functionality e.g. a network card may 'drop' in-coming packets or a serial line may 'drop' received characters.

### Accelerator Mode

The accelerator architecture offers *predictable* and consistent interrupt *dispatch latency* resulting in higher performance for large PCI configurations. *Predictability* is key in most real-time applications.

The corresponding accelerator interrupt *dispatch latency* is always two Bus read cycles.

## Physical Implementation

This paper is not intended to imply any particular physical implementation. The generic functionality for a PICMG application can be implemented either as an ISA or PCI based device, however it could also be supported in alternative bus architectures.

## Summary

Standard motherboard implementations provide PCI interrupt binding [for the firmware/BIOS] decode in the physical etch routing.

This binding structure becomes congested when additional [bridged] PCI buses are implemented in the system. This means that the polling and decode of an interrupt request can be significantly less than optimal. The interrupt latency is also unpredictable.

The proposed interrupt accelerator design described in this paper results in the *predictable*, repeatable [consistent] and improved interrupt dispatch latency, *key for real-time applications*.

## Acknowledgements

The DMCC program was the co-operative effort of a large number of engineers whose initiatives have contributed to this paper. Thanks in particular to Alan Milne, Sean McGrane, Robin Alexander, Vikas Sontakke and John Lenthall, all of whom have worked within the E&Rt engineering design team to ensure the successful product implementation of the concepts detailed in this paper.

## References

1. PCI Local Bus Specification, Revision 2.0
2. PICMG, PCI-ISA Card Edge Connector Proposal for Single Board Computers, Revision 2.0

3. PICMG, PCI-PCI Bridge Board Edge Connector Proposal for Single Board Computers, Revision 1.01
4. PCI to PCI Bridge Architecture Specification, Revision 1.0

## **Author**

Ross Armstrong, the Project Leader for the Digital Modular Computing Components [DMCC] program, is a principal hardware design engineer with Digital's Embedded and Real-time [E&Rt] engineering organisation. Ross holds a B.Sc. (Eng.) Hon's from Aberdeen University and a joint Master of Technology Management [M.T.M.] from Strathclyde and Heriot Watt Universities.

## DCM'S PCI - TO - PCI BRIDGE SOLUTION

Kamal Mansharamani  
DCM DATASYSTEMS  
Vikrant Tower  
4, Rajendra Place  
New Delhi - 110008, India  
91-11-5737397/5755731(fax)  
email: dcmds@giasdl01.vsnl.net.in

### 1.0 *Abstract*

The PCI bus has now become a defacto industry standard. Its high bandwidth makes it very attractive for high performance server and multimedia applications. However, the high speed of the bus also puts a limitation on its expansion capability. A PCI system can have only three to four expansion slots. This can become a severe limitation for contemporary applications.

The PCI bus can support a load of upto ten devices. Each device on the motherboard is one load, while each device on an add-on card is two loads. Since, each motherboard has typically two to three on board PCI devices, this means that a motherboard can have only three to four expansion slots. The PCI bus also imposes limitations on the add-on cards. Each add-on card can present only one PCI load to the bus. This can also severely limit the functionality that can be offered on PCI add-on cards. Currently, there are a variety of PCI chips in the market including SCSI, Ethernet, VGA etc. However, only one of these devices can be present on the PCI add-on card.

There is clearly a need to enhance the functionality of PCI systems both in terms of the expansion capability of motherboards, as well as the functionality of the add-on cards. The PCI-to-PCI bridge provides a solution for both these requirements

This paper will discuss the technical issues encountered in the design of a high performance PCI-to-PCI bridge chip. Thereafter, DCM's PCI-to-PCI bridge chip and its architecture will be presented.

### 2.0 *Introduction*

A PCI-to-PCI bridge connects between two PCI buses and allows expansion of the PCI bus. A PCI-to-PCI bridge chip expands the electrical capacity of the PCI bus. It can be connected to the PCI bus closest to the Host CPU and used to increase the expansion capability of the system. In fact, multiple PCI bridges can be connected on the bus to provide theoretically unlimited expansion capability (Fig 1). The PCI-to-PCI bridge can also be used to increase the functionality of the add-on cards. One can build Multi-function combo cards (Fig 2) with functions like Ethernet, SCSI and Graphics support. It is also possible to build Multi channel cards like Multi channel SCSI or Multi port Ethernet boards.

In fact, a PCI-to-PCI bridge can go much beyond increasing the electrical loading capability of the systems. A bridge can also isolate the traffic on both the sides. It can also allow concurrent operations on the primary as well as the secondary bus (Fig 3). The transactions between master #0 and target #0 on the primary bus can go on concurrently with the transactions between master #1 and target #1 on the secondary bus. This can tremendously increase the bandwidth of the system. For example, if we have a graphics and a video device on the secondary PCI bus then the transactions between the two can take place without crossing the PCI bridge. The bridge chip can also boost the performance of the system by incorporating features like posted write, read pre-fetch etc.



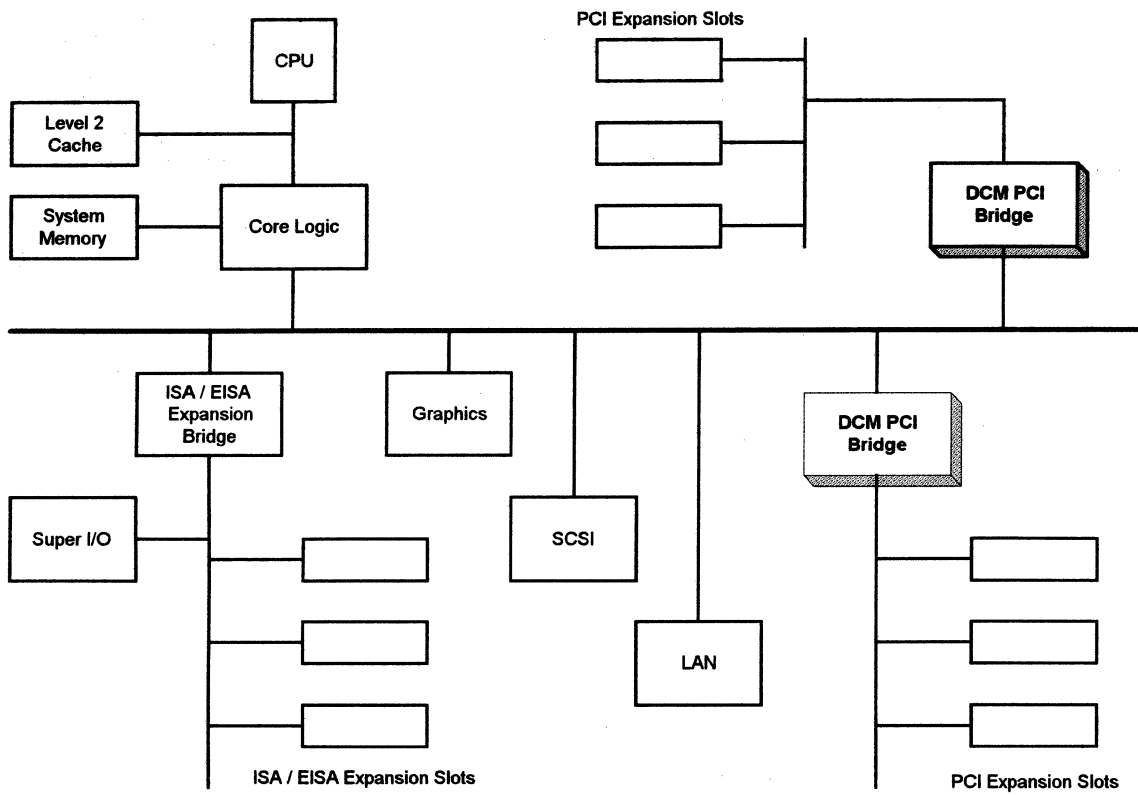


Fig. 1

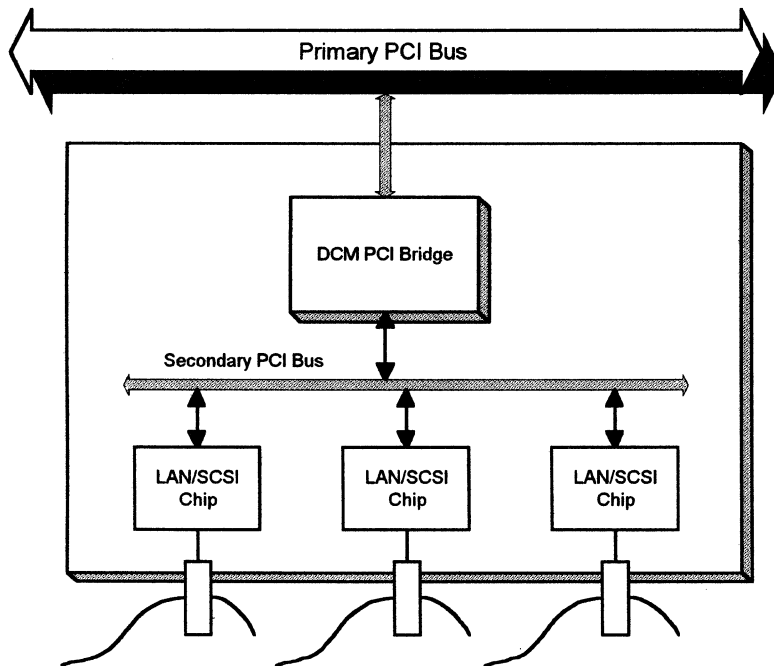


Fig. 2

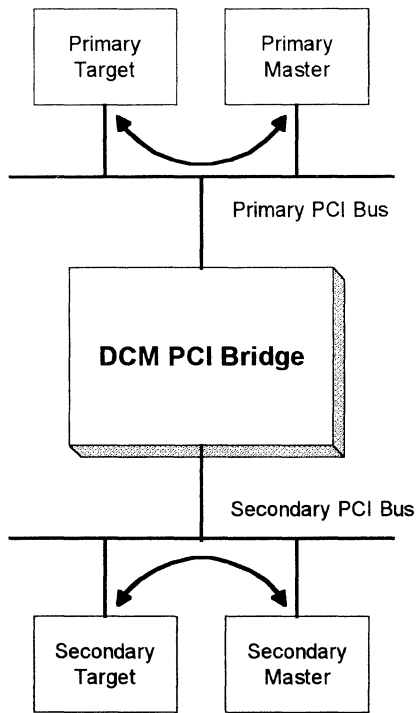


Fig. 3

### 3.0 *Design Issues*

Let us now look at the issues involved in the design of a PCI-to-PCI bridge. The design of the bridge chip would have to necessarily look at the following issues:-

- Transparency
- Compatibility
- Compliance
- Performance

#### 3.1 Transparency

The PCI-to-PCI bridge would have to be totally transparent to the system. It should make no difference whether a device is connected before the bridge or after the bridge. The system should not need any device drivers to support the bridge chip.

#### 3.2 Compatibility

A major compatibility issue in terms of PCI-to-PCI bridge design is the issue related to support of VGA compatible devices. These issues are mainly related to ISA compatible addressing and palette snooping. In order to support the VGA device downstream of a PCI-to-PCI bridge, the bridge must have the capability to be configured to recognize the ISA compatible addresses used by the VGA devices. The bridge must also support configurations where a graphics device downstream of a bridge needs to snoop VGA palette accesses.

#### 3.3 Compliance

Some of the major compliance requirements for a PCI - to - PCI bridge chip are listed below.

3.3.1 The PCI-to-PCI bridge must be compliant with the PCI Local Bus Specification. This would essentially mean the following:-

3.3.1.1 The bridge must adhere to the electrical loading limits for all the PCI signals. This means that the PCI bridge is limited to present a single load per connection.

3.3.1.2 The bridge must maintain data coherency and consistency when transactions cross the bridge in either direction.

3.3.2 The bridge must comply with the current PCI-to-PCI Bridge Architecture Specification. Some of the required capabilities are as follows:-

3.3.2.1 The bridge must support configuration space conforming to PCI-to-PCI bridge header format.

3.3.2.2 The bridge must support memory mapped I/O address space.

3.3.2.3 The bridge must have hierarchical configuration support.

3.3.3 The bridge designs which support arbiter on the secondary bus must be designed to prevent deadlocks.

### 3.4 **Performance**

Performance is a major consideration in the design of a PCI-to-PCI bridge chip. In fact, it is possible to significantly improve the performance of a system by properly designing the bridge chip. Some of the design elements which can boost performance are as follows :-

#### - **High Speed FIFO Design**

The ability of the FIFO to support transfers with every PCI clock in either direction is a very crucial parameter in the design of the bridge chips. The FIFO needs to be designed in a way to support 1-1-1 transfers concurrently in either direction.

#### - **Increased Buffer Size**

The size of the FIFO buffer also plays an important role in determining the performance of the bridge chip. The FIFO should have sufficient depth to be able to support sustained transfers in either direction. The FIFO design which includes the FIFO architecture and the FIFO depth plays a very crucial role in determining the performance of the bridge.

#### - **Support for Delayed Transactions**

Support for Delayed transactions is one of the important features which has been added in the PCI Revision 2.1 specifications. Delayed transactions are used normally while accessing slow devices. One of the major advantages of delayed transactions is that the bus is not held in wait states while an access is being completed on a slow device. Delayed transactions are used for all those commands which can complete on the target bus before completing on the originating bus. A delayed transaction is composed of three phases :

1. Request by the master
2. Completion of the request by the target
3. Completion of the request by the master

During the first phase, the master would generate a request on the primary bus. The bridge would decode the cycle, latch the

information required to complete the access and terminate with a retry.

During the second phase the bridge would independently complete the access on the secondary bus. The bridge would store the data and the status pertaining to the delayed request.

During the third phase, the master would successfully arbitrate for the bus, acquire it, and reissue the original request. The bridge would decode the cycle and provide to the master the stored data and status.

#### - **Multiple Delayed Transactions**

The bridge chips also have the capability to support multiple delayed transactions to improve the system performance and also to meet the initial latency requirements. The most important requirement for supporting multiple delayed transactions is that the ordering of the transactions be maintained and the deadlocks be avoided.

#### - **Transaction Ordering**

The rules on transaction ordering accomplish three things.

First of all, they ensure ordering of write results, which means that ordering is maintained across the system.

Secondly, they allow for posting of transactions which improves system performance.

Thirdly, the rules also prevent bus deadlock conditions.

#### - **Combining, Merging, and Collapsing**

Bridges can also convert a transaction with single or multiple data phases into a larger transaction to optimize data transfer on the PCI bus. The various terms used for this are defined as: Combining, Collapsing and Merging.

#### **Combining**

Combining takes place whenever sequential memory write transactions are combined using a single PCI transaction by using linear burst ordering. Combining takes place within the bridge and the target sees the data in the same order in which the originating master generated it.

## Byte Merging

Byte merging occurs whenever memory writes consisting of bytes and words are combined into DWORDS. Byte merging should only be done when the bytes in the data phase are within the prefetchable address range.

## Cacheline Merging

This occurs whenever a sequence of memory writes are merged into a single cacheline.

## Collapsing

Collapsing occurs whenever a sequence of memory writes to the same location are collapsed into a single bus transaction. Collapsing is normally not permitted in PCI bridges except in very specific conditions.

## 4.0 DCM'S PCI - to - PCI Bridge Solution

DCM DATASYSTEMS has designed a high performance PCI - to - PCI bridge chip. It supports the following features:-

### 4.1 Features

- Supports two 32-bit PCI Rev 2.1 buses
  - Supports Delayed Transactions
  - Stores upto three Delayed Transactions
  - Supports maximum clock speed of 33 MHz
  - Implements PCI Rev 2.1 Drivers
  - Provides concurrent primary and secondary bus operation
  - Conditionally forwards the following transactions :
    - Memory read and write transactions in either direction
    - I/O read and write transactions in either direction
- Configuration read and write transactions in the downstream direction
  - Configuration write transactions to special cycles in either direction
  - Supports memory transaction filtering through two programmable memory address regions - one prefetchable and one non-prefetchable
  - Supports 64-bit addressing
  - Supports read prefetching for memory read transactions
  - Provides extensive buffering for - writes and reads - both in Up Stream and Down Stream directions
  - Provides upto 88 bytes of write posting for memory write transactions
  - Provides upto 72 bytes of read data buffering
  - Provide I/O transaction filtering through one programmable memory I/O address region
  - Provides ISA mode for I/O transaction filtering
  - Provides two programmable video graphics adapter (VGA) bits that support forwarding of VGA memory and I/O addresses, or forwarding of VGA palette I/O writes
  - Provides master latency timers and target wait timers, for each PCI interface, which limit the amount of latency on either bus
  - Provides concurrent resource lock operation
  - Propagates locks across the Bridge
  - Provides five secondary PCI bus clock outputs
  - Provides the following optional central functions :

- Programmable rotation arbitration function supporting upto six secondary bus masters
- Secondary PCI bus parking at the Bridge
- Supports Perr and Serr signals through error checking functionality

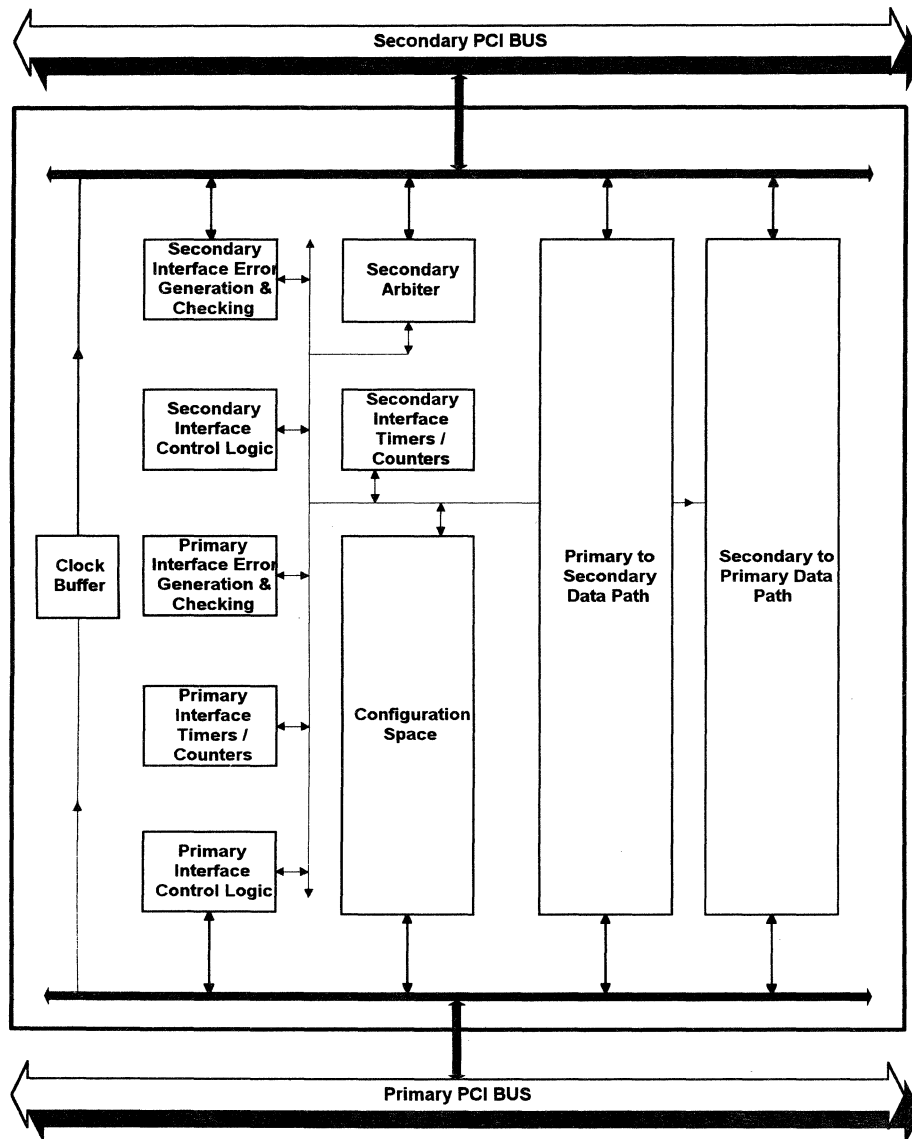


Fig. 4

#### 4.2 Architecture

The block diagram of DCM's PCI-to-PCI bridge chip is shown in Fig-4. It consists of the following major blocks:-

#### Primary to Secondary Data Path

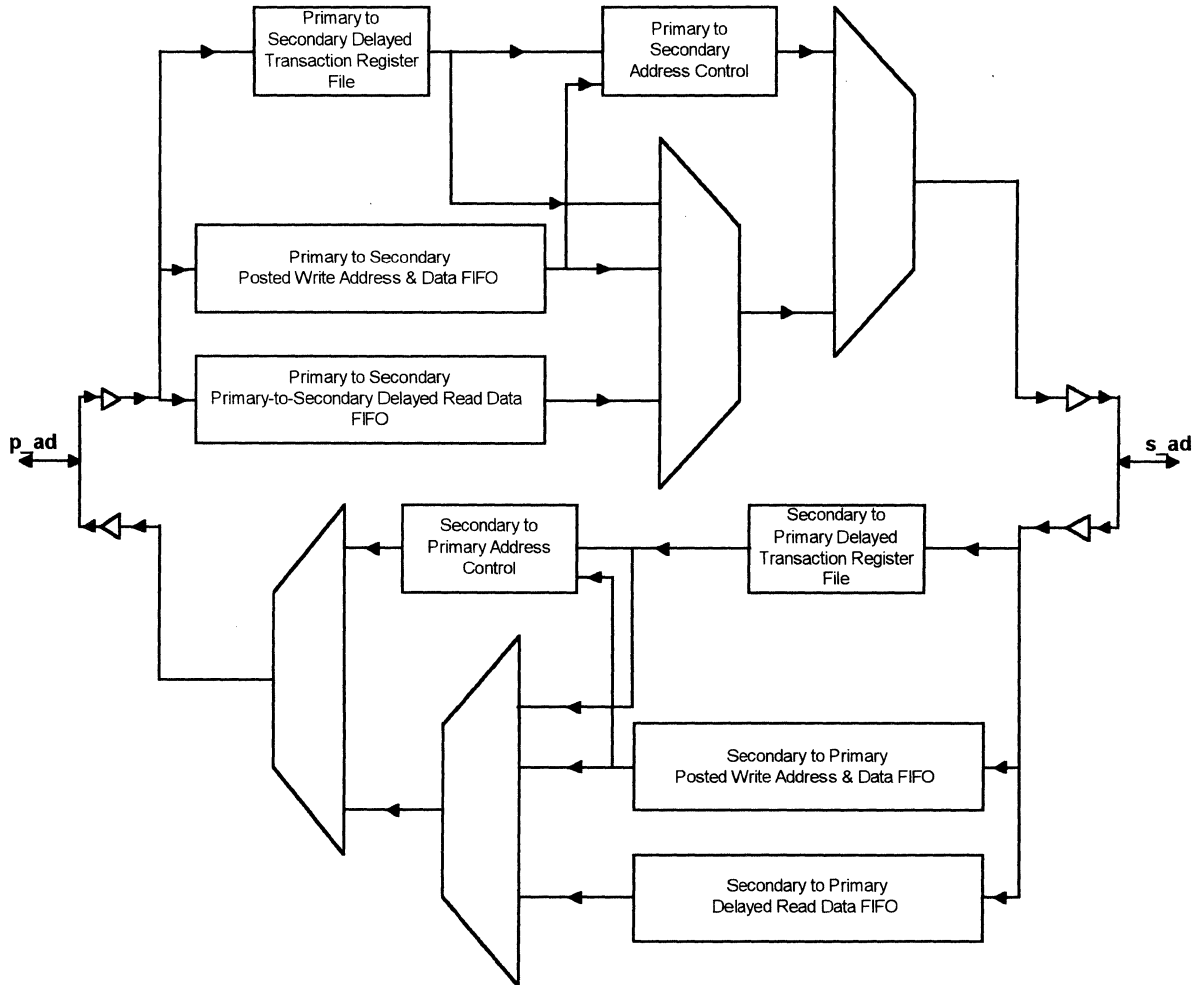
This block contains the logic for driving the data received on the Primary bus onto the Secondary

bus. It also contains the logic and the buffers for write posting as well as read pre-fetching.

### Secondary to Primary Data Path

This block drives the data received on the Secondary bus onto the primary bus. This block also contains the logic as well as the buffers for write posting as well as read pre-fetching.

Fig 5 shows an exploded version of the primary to secondary and secondary to primary data paths. Each data path supports logic element and FIFOs to support posted write, read pre-fetch and support for delayed transactions. The FIFOs are designed in such a way to support 1-1 transfers in both the directions.



**Fig. 5**

### Primary Control

Primary Control consists of the control logic and the state machines to handle all the transactions

initiated on the primary interface. It consists of the following logic elements:-

- Primary Interface Control Logic
- Primary Interface Timers/Counters

- Primary Interface Error generation and checking

### **Secondary Control**

Secondary Control is used to handle all the transactions initiated on the secondary bus. It also consists of an arbiter for arbitrating the requests on the secondary bus. It consists of the following elements:-

- Secondary Interface Control Logic
- Secondary Interface Timers/Counters
- Secondary Interface Error generation and checking
- Secondary arbiter

### **Configuration Space**

Configuration space consists of the configuration registers for the bridge.

## **5.0 Conclusion**

The high bandwidth offered by the PCI systems coupled with the availability of high performance CPU's like the Intel Pentium Pro, DEC Alpha and the IBM Power PC has now made it possible to configure high performance systems for server as well as Multi-media applications. The only limitation on the PCI bus is its expansion capability. A typical PCI system would have only three to four expansion slots. In addition, the PCI add-on cards can have only one PCI device on the board. This creates a severe limitation on the expansion capability of PCI motherboards and the functionality of PCI add-on boards. The PCI-to-PCI bridge offers a solution to both these problems. In addition, properly designed PCI -to-PCI bridges can significantly boost the performance of PCI systems. The tremendous potential promised by PCI systems can be fully realized by the PCI-to-PCI bridge chips.

## PC-DMA AND PCI: NEW OPEN STANDARD BLENDS BOTH

Dwight D. Riley  
Systems Technology Development  
Compaq Computer Corporation  
MS 100505  
20555 S.H. 249  
Houston, TX 77070-2698  
Fax (713) 518-0025  
e-mail: driley@bangate.compaq.com

### **ABSTRACT**

A serious impediment to the development of PCI-only systems is the requirement for compatibility with existing PC DMA devices like floppy disk controllers, 16-bit PC Cards (PCMCIA cards), and sound cards. On December 1, 1995, seven system and chipset manufacturers published a new open standard, *Distributed DMA Support for PCI Systems*, which defines a DMA architecture that is software compatible with PC DMA controllers yet works without using any sideband signals.

### **INTRODUCTION**

The PCI bus has become the standard peripheral bus of choice, taking over the market quickly since its introduction by Intel in 1992. It is used in all kinds of computers, from supercomputers to portables. However, despite its numerous successes it has been unable to replace the traditional PC expansion buses (ISA, EISA, and Micro-Channel). These buses have not disappeared because the PCI bus was designed assuming an expansion bus would exist to provide PC legacy functions such as DMA.

DMA was first introduced on the ISA bus in the original PC in 1981. DMA provided a mechanism for the processor to off-load the work of moving I/O data between an I/O device and system memory. The microprocessor still had to configure the block data transfer between the I/O device and memory, but the actual data transfer and its termination were handled solely by the DMA controller. Not only did it reduce the processor's workload, it also increased overall bus bandwidth. This increase in bandwidth was a result of the DMA Controller's ability to perform a "fly-by" transfer: an I/O read done simultaneously with a memory write, or an I/O write done simultaneously with a memory read. To perform the same task, the processor would have to run two bus cycles.

With the wide acceptance of the PCI bus, there has been a natural migration of ISA devices to PCI. This migration, however, has not been able to include legacy PC DMA devices without losing software compatibility.

### **THE PROBLEM**

The legacy DMA Controller was implemented using two Intel 8237s, each of which provided four separate DMA channels. The 8237s were connected in a cascaded configuration, providing a total of seven DMA Channels as illustrated in Figure 1. The first 8237 provided support for channels 0 through 3, while the second 8237 provided support for channels 5 through 7. Channel 4 of the second 8237 was used to link the two 8237s together.

Bringing PC DMA to the PCI bus has a single fundamental requirement that must be met: **the need to maintain the legacy 8237 DMA programming model**. This requires a solution that retains the I/O register interface provided by the pair of 8237s. It also requires a solution that provides a bus mastering service to replace the DMA controller functionality - mimicking the old DRQ and DACK# protocol and running I/O and memory cycles on behalf of the DMA device. In addition, it requires that the shared, multi-channel DMA registers be isolated so that the DMA channels could be separated on the PCI bus. This is because the PCI bus does not allow multiple devices to drive different data bits for a single access.



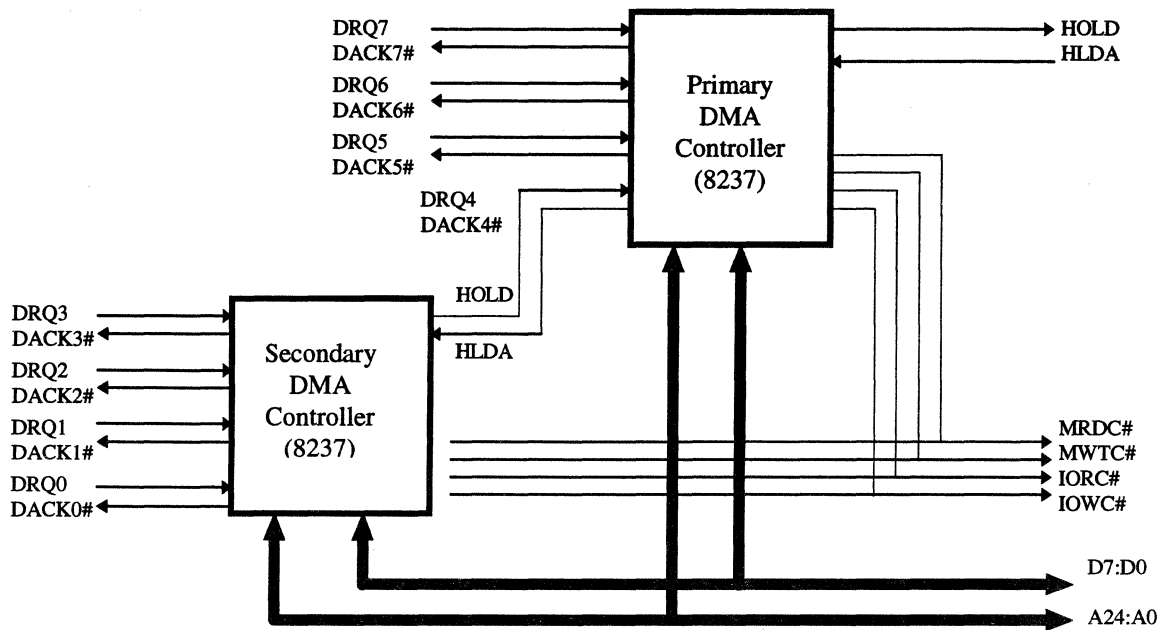


Figure 1: Legacy DMA Controller. ISA systems contained two cascaded 8237s.

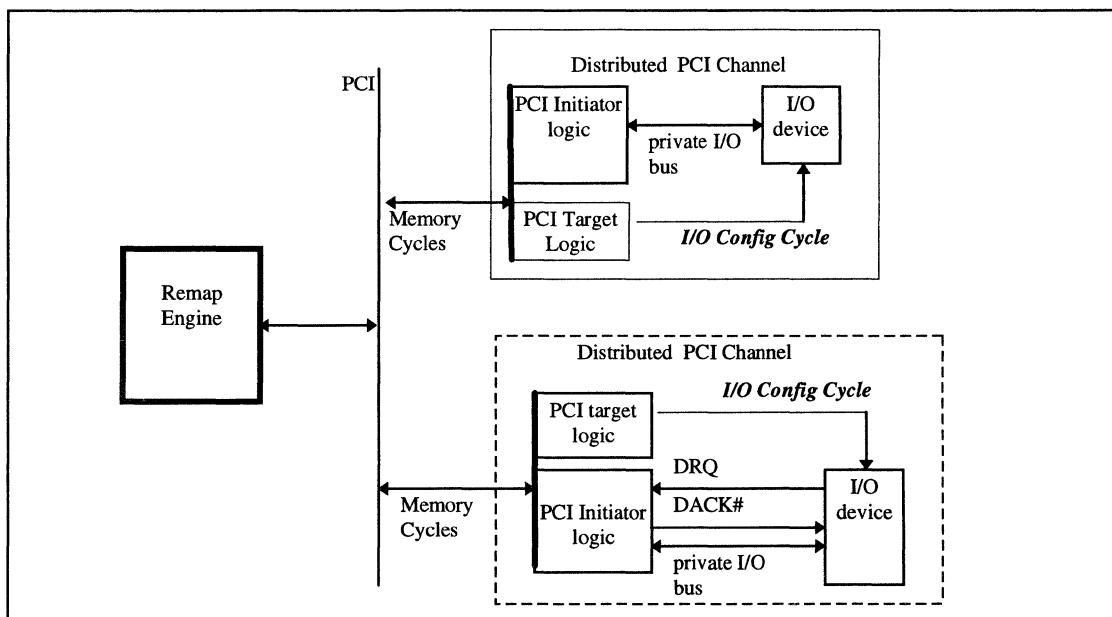
The Distributed DMA specification resolves all these issues and breaks the legacy hardware model of a centralized DMA controller, building instead on the fundamentals of the PCI bus mastering model. There are several advantages to the Distributed DMA approach:

1. It provides the ability to separate and isolate the various DMA channels so that they can exist singularly in the PCI devices, tightly coupled to their I/O devices. The result is the removal of the DMA DRQ/DACK# signals from the PCI connection to a private I/O bus that exists behind the Distributed DMA Channel.
2. It does not require any sideband signals, thereby complying with the existing PCI bus specification. This scheme is flexible enough to work from a generic PCI plug-in slot.
3. It is fully PCI compatible even across PCI bridges, an attractive feature for PCI-only systems that use at least one PCI to PCI bridge to increase PCI bus connectivity.
4. It allows for mobile PC docking across the PCI bus with no additional sideband signals.
5. It removes the long I/O cycle portion of the transfer (which takes up to 1  $\mu$ s on ISA) from the PCI bus. PCI bandwidth is only used for the faster system memory access portion of the transfer.
6. Distributed DMA can also co-exist with an existing legacy DMA Controller on the standard expansion bus. This allows for upgrading existing PCI systems to support Distributed DMA.
7. New drivers can be written to exploit PCI bus performance by communicating directly to the Distributed DMA Channel interfaces.
8. Distributed DMA also provides for 32-bit extensions to the legacy DMA Controller's programming model, thereby facilitating porting 32-bit DMA devices from EISA and Micro-Channel.

## THE SOLUTION

There are two fundamental ideas to implementing Distributed DMA. The first is to isolate the DMA channels as Distributed PCI Channels. The second is to map all legacy DMA accesses to these new isolated DMA channels via a Remap Engine. Both the Distributed PCI Channels and the Remap Engine are illustrated in Figure 2. These fundamental ideas can be implemented using three main components:

1. The *Distributed PCI Channel's PCI Target Logic*. A standard PCI target responsible for configuring the Distributed PCI Channel and the I/O device.
2. The *Distributed PCI Channel's PCI Initiator Logic*. A standard PCI bus master responsible for moving data between memory and the I/O device.
3. The *Distributed DMA Remap Engine*. Logic responsible for maintaining legacy software compatibility.



**Figure 2 -- A Distributed DMA System.** This diagram illustrates the Remap Engine block and two Distributed PCI Channels. The top illustrates an integrated design while the bottom illustrates a more traditional protocol using DRQ/DACK#.

### The Distributed PCI Channel

The Distributed PCI Channels combine specially defined target and initiator logic as illustrated in Figure 2. One or more original I/O devices are connected to these modules via some private interfaces. The interfaces could be a full legacy bus (e.g. to implement an ISA or PCMCIA bridge) or some private interface contained within a chip (e.g. to implement an audio card).

#### *Distributed PCI Channel Target -- PCI Target Logic*

The first major component of a Distributed DMA system, as shown in Figure 2, is the PCI Target Logic of the Distributed PCI Channel. It contains configuration registers for the DMA transfers and the mechanism for programming the I/O device. Like any PCI device using I/O addressing, it contains a Configuration Space Header defining its I/O base addresses. This is used to set up a separate 16-byte I/O window of registers for that channel, as

listed in Table 1. The register definitions are similar to those of the legacy DMA Controller, but they only operate on one channel. In addition, the programming interface is also extended to support 32-bit addressing.

**Table 1 -- Distributed PCI Channel's I/O Map. Arrows show legacy DMA channel 0 I/O space being mapped into the new Distributed PCI Channel I/O space**

Secondary 8237 (Four Channels)				Distributed PCI Channel I/O Map			
Addr.	R/W	Channel	Register Name	Register Name	R/W	Offset	
0000h	W	0	Base Address [7:0] & [15:8]	Base Address [7:0]	W	00h	
0000h	R	0	Current Address [7:0] & [15:8]	Current Address [7:0]	R	00h	
0001h	W	0	Base Count [7:0] & [15:8]	Base Address [15:8]	W	01h	
0001h	R	0	Current Count [7:0] & [15:8]	Current Address [15:8]	R	01h	
0002h	W	1	Base Address [7:0] & [15:8]	Base Address [23:16]	W	02h	
0002h	R	1	Current Address [7:0] & [15:8]	Current Address [23:16]	R	02h	
0003h	W	1	Base Count [7:0] & [15:8]	*Base Address [31:24]	W	03h	
0003h	R	1	Current Count [7:0] & [15:8]	*Current Address [31:24]	R	03h	
0004h	W	2	Base Address [7:0] & [15:8]	Base Word Count [7:0]	W	04h	
0004h	R	2	Current Address [7:0] & [15:8]	Current Word Count [7:0]	R	04h	
0005h	W	2	Base Count [7:0] & [15:8]	Base Word Count [15:8]	W	05h	
0005h	R	2	Current Count [7:0] & [15:8]	Current Word Count [15:8]	R	05h	
0006h	W	3	Base Address [7:0] & [15:8]	*Base Word Count [23:16]	W	06h	
0006h	R	3	Current Address [7:0] & [15:8]	*Current Word Count [23:16]	R	06h	
0007h	R/W	3	Base Count [7:0] & [15:8]	Reserved	W	07h	
0007h	R/W	3	Current Count [7:0] & [15:8]	Reserved	R	07h	
0008h	W	0,1,2,3	Command	Command	W	08h	
0008h	R	0,1,2,3	Status	Status	R	08h	
0009h	W	0,1,2,3	Request	Request	W	09h	
000Ah	R	0,1,2,3	Mask	Reserved	R	0Ah	
000Bh	W	0,1,2,3	Mode	Mode	W	0Bh	
000Ch	W	0,1,2,3	Clear Byte Pointer	Reserved	W	0Ch	
000Dh	W	0,1,2,3	Master Clear	Master Clear	W	0Dh	
000Dh	R	N/A	Temporary	Reserved	N/A	0Eh	
000Eh	W	0,1,2,3	Clear Mask	Mask	R/W	0Fh	
000Fh	R/W	0,1,2,3	Multi-Channel Mask				
0087h	R/W	0	Low Mem. Page [23:16]				
0083h	R/W	1	Low Mem. Page [23:16]				
0081h	R/W	2	Low Mem. Page [23:16]				
0082h	R/W	3	Low Mem. Page [23:16]				

This new programming model is designed for easy legacy DMA Controller compatibility, and allows new Distributed PCI Channel driver software to borrow existing 8237 legacy code. This reusability of existing legacy DMA drivers to drive what is effectively a new standard programming model for PCI masters has several benefits:

1. It builds on an existing knowledge base resulting in a greater pool of people, experts, to choose from when it comes time to port existing PC DMA drivers to true native PCI bus masters' drivers.
2. The majority of the changes required to write a Distributed PCI Channel driver only require changing the address location of the I/O device. That is, replacing the existing fixed legacy 8237 I/O addresses to a new set of I/O addresses that are offset from a programmable base.

As shown in Table 1, the DMA registers *Command*, *Status*, *Request*, *Mode* and *Master Clear* are duplicated in function by each Distributed PCI Channel at offsets 08h, 08h, 09h, 0Bh, and 0Dh, respectively. The *Mask*, *Clear Mask*, and *Multi-Channel Mask* registers are all mapped into a single register called *Mask* at offset 0Fh. Unlike the original DMA controller registers, these are restricted to only affect a single channel.

The *Address* low and high bytes are directly accessible (no Byte Pointer controlling access through one register), and the *Low Page* registers are included to provide 24 bits of address at offsets 00h, 01h and 02h. Similarly, the *Word Count* low and high bytes are separated and mapped to offsets 04h and 05h. In addition to the legacy addressing support, bits were added to optionally allow 32-bit addresses and 24-bit word counts. These extensions allow for 4 GB of addressable memory space and a transfer count as high as 16 MB.

The *Clear Byte Pointer* register is not needed since the *Address* and *Word Count* low and high bytes are individually accessible. The *Temporary* register is also not needed, because it does not affect channel configuration. The DMA Remap Engine will provide Temporary register support.

### ***Configuring the DMA I/O Device***

The Distributed DMA specification does not control how to configure the DMA I/O device that is serviced by the Distributed PCI Channel. However, the standard PCI configuration base address registers should provide the necessary I/O space required to configure the I/O device that is serviced by a Distributed PCI Channel.

Some legacy DMA I/O devices, however, must carry their fixed legacy I/O addresses to the PCI bus. As such these devices must be placed on the primary PCI bus and positively decode these addresses in order to maintain legacy compatibility. For example, system audio compatibility requires legacy I/O address 0220h and 0240h. If this device is placed on the primary PCI bus it can positively decode these addresses before the standard expansion bus bridge claims the cycles.

Designs that must continue support of certain legacy I/O addresses may be required to support both the PCI programmable base address register scheme as well as a fixed legacy I/O addressing mode. This allows the design to be initialized as a generic PCI master with a relocatable I/O address range as its default. If the system needs to support the legacy mode, these devices can be reconfigured to use their fixed legacy I/O address decode ranges. The system software must resolve I/O conflicts on PCI when operating these devices in their legacy I/O decode modes.

### ***Distributed Channel Initiator -- PCI Initiator Logic***

The second major component of a Distributed DMA system, as shown in Figure 2, is the PCI Initiator Logic of the Distributed PCI Channel. This PCI Initiator Logic is responsible for servicing the DMA I/O device. For example, an I/O device requests a DMA transfer by signaling the Initiator Logic. If programmed correctly (with the channel enabled and unmasked), the Initiator Logic responds and begins PCI memory read or write transfers on behalf of the I/O device, using the programmed address, word count, and mode. If the I/O device is an ISA-based design, the request is made with a DRQ signal, the response is expected on a DACK# signal, and the transfers to the I/O device should be ISA I/O reads and writes. The I/O device may have some other interface; the communication protocol to the I/O device is undefined by the Distributed DMA specification.

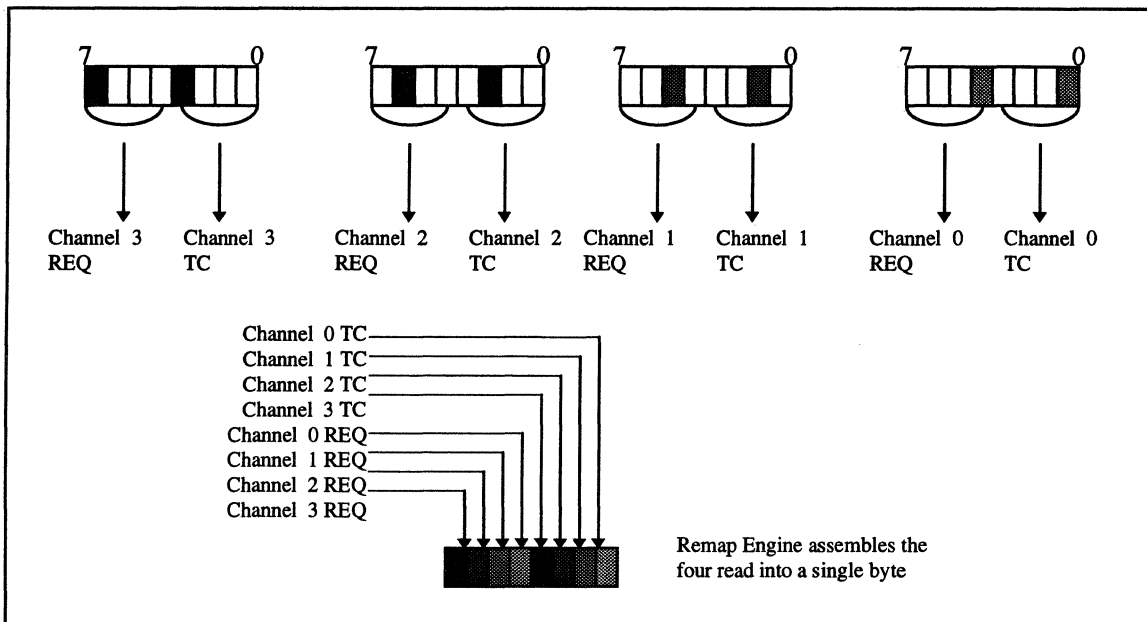
To increase performance, the PCI Initiator Logic can buffer and later burst write data. It can also read a full 32 bits at a time or perform burst read-ahead to increase read performance. Note that PCI bandwidth is only used for the memory transfer portion of the transfer. The I/O transfer (if it even exists) is not seen on the PCI bus.

### **The Remap Engine**

The third major component of a Distributed DMA system, as shown in Figure 2, is the Remap Engine. Its function is to provide the appearance to the software that the channels are linked together in an identical fashion to the legacy DMA Controller programming model, even though the Distributed PCI Channels are in fact isolated.

The Remap Engine accomplishes this by capturing the PCI cycles which would otherwise access an 8237 register and spawning other PCI cycles that actually access the Distributed PCI Channels. These spawned cycles are used by the Remap Engine to update or gather status from the Distributed PCI Channels when a legacy DMA register is accessed.

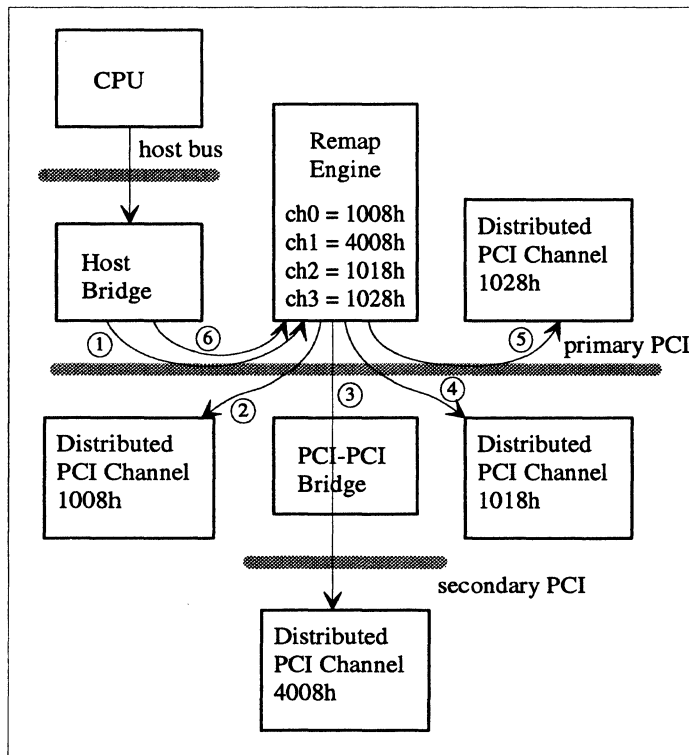
For example, Figure 3 illustrates that when the *Status* register at I/O address 0008h is accessed, four cycles are spawned. Each of the Distributed PCI Channels returns two bits of status: the terminal count (TC) replicated on bits 0-3 and channel request (REQ) replicated on bits 4-7.



**Figure 3 -- Distributed DMA status read data merge. Shows the status read data merge from four Distributed PCI Channels.**

The actual sequence is illustrated in Figure 4, with the following steps:

- Step 1 -- The initial PCI read of the legacy *Status* register occurs on the PCI bus. As a direct result of the processor's initial request, the Remap Engine will respond with a PCI delayed transaction reply (a Retry).
- Step 2 -- The Remap Engine will then arbitrate for the PCI bus, and spawn a PCI read access to Distributed PCI Channel at I/O location 1008h (which was previously assigned to channel 0). The return data is then stored in a temporary holding register.
- Step 3 -- The Remap Engine makes another request and runs a PCI read from the Distributed PCI Channel at I/O location 4008h (previously assigned to channel 1). This Distributed PCI Channel is illustrated as existing on a secondary PCI bus behind a PCI-to-PCI bridge. The return data, consisting of the Terminal Count and the Request Bit, is then merged into the temporary holding register as illustrated in Figure 3 using the logical channel assignment as a guide to the correct bit positions.
- Steps 4 & 5 -- The Remap Engine accesses Distributed PCI Channels at I/O locations 1018h, and 1028h, assigned to channels 2 and 3 respectively. As each read completes, that *Status* data is merged into the temporary holding register, using the channel's logical assignment as a guide to the correct bit positions. The result is a reassembled multi-channel legacy *Status* register in the Remap Engine.
- Step 6 -- When the requesting PCI agent repeats the original I/O read transaction, the content of the temporary holding register is returned as the legacy *Status* register, thereby completing the delayed transaction cycle.



**Figure 4 -- Status read, with four spawned cycles**

Note that the Remap Engine is free to be placed anywhere on the primary PCI bus using delayed transactions to hide the I/O mapping. For better performance, the Remap Engine can be designed into the host bridge, eliminating the need for the retry cycles (step 1 and step 6 in the above example).

The Distributed DMA specification recommends two techniques for configuring the Remap Engine with logical channel assignments for the Distributed PCI Channel. The two techniques are:

1. The Remap Engine uses a single base address from which 128 bytes of contiguous I/O space are reserved for the Distributed PCI Channels. This 128 byte of I/O space is then divided into 16-byte I/O blocks. Each of the 16-byte blocks corresponds to a different channel, with offsets 00h through 0Fh for Channel 0, offset 10h through 1Fh for channel 1, etc. A 16-byte hole is left for channel 4, which is not usable. The I/O base address of each Distributed PCI Channel must be programmed to make that channel's I/O map occupy the appropriate 16-byte block.
2. The Remap Engine has separate base address registers for each channel, thus removing the restriction that the I/O spaces for the Distributed PCI Channels be in consecutive 16-bytes blocks. This technique allows for a more generic solution that can facilitate the PCI to PCI bridge requirement that I/O space assignments must be done in 4 Kbyte blocks.

Table 2 and Table 3 show the legacy I/O address mapping for channels 0-3, along with the number of spawned cycles that the Remap Engine issues for each legacy access. Channels 5-7 are similar, with the maximum number of spawned cycles being 3 instead of 4. This is because channel 4 is used only for providing cascading support for channels 0-3; the channel is never used for I/O devices. This table assumes that Channel 4's cascading effects are ignored.

**Table 2 -- Spawned cycles for the Secondary DMA channels 0-3 *Control* register access.**

Legacy DMA 0-3 Address	R/W	Register Name	Max. # of Spawned Cycles	Distributed DMA 0-3 offset
0008h	W	Command	4 writes	08h
0008h	R	Status	4 reads	08h
0009h	W	Request	1 write	09h
000Ah	W	Mask	1 write	0Fh
000Bh	W	Mode	1 write	0Bh
000Ch	W	Clear Byte Pointer	0 write	N/A
000Dh	W	Master Clear	4 writes	0Dh
000Dh	R	Temporary	0 reads	N/A
000Eh	W	Clear Mask	4 writes	0Fh
000Fh	R/W	Multi-Channel Mask	4 R/Ws	0Fh

**Table 3-- Spawned cycles for the Secondary DMA channels 0-3 *Base Address and Word Count* register access.**

Legacy DMA Ch0	Legacy DMA Ch1	Legacy DMA Ch2	Legacy DMA Ch3	R/W	Register Name	Max. # of Spawned Cycles	Distributed DMA 0-3 offset
0000h	0002h	0004h	0006h	W/R	Base Address Low [7:0]	1	00h
0000h	0002h	0004h	0006h	W/R	Base Address High [15:8]	1	01h
0087h	0083h	0081h	0082h	W/R	Low Mem. Page [23:16]	1	02h
0001h	0003h	0005h	0007h	W/R	Word Count Low [7:0]	1	04h
0001h	0003h	0005h	0007h	W/R	Word Count High [15:8]	1	05h

#### Supporting the Cascading Effect of Channel 4

The *Distributed DMA Support for PCI System* revision 6.0 specification does not address supporting the cascading effect of channel 4 on channels 0-3. In the legacy DMA Controller channel 4 is not a usable DMA channel, and must be configured in cascade mode at system initialization. However, the channel 4 registers are still accessible, and some can be modified without reconfiguring the channel's cascade mode. This means channel 4 settings can have an effect on channels 0, 1, 2, and 3.

For example, if channel 4 is masked, Secondary DMA channels 0-3 are also effectively masked. Taking advantage of this global effect, though unorthodox, is still very much "PC legal." The following legacy Primary DMA registers can cascade onto channels 0-3:

- *Command Register* (Port 00D0h). A write to disable channels 4-7 must also disable channels 0-3. A write to enable channels 4-7 must also enable channels 0-3 if they were disabled only due to a previous write to disable channels 4-7.
- *Single Mask Bit* (Port 00D4h). A write to mask channel 4 must also mask channels 0-3.
- *Multi-Channel Mask Bit* (Port 00DEh). A write which masks channel 4 must also mask channels 0-3.
- *Master Clear* (Port 00DAh). A write access masks and enables channels 4-7. This transaction must be handled as if a combination of *Command* and *Multi-Channel Mask* writes occurred.

Supporting this cascading effect of channel 4 on channels 0-3 requires the Remap Engine use a more complicated mapping algorithm for updating the Distributed PCI Channels.

First, the Remap Engine needs to track the Primary and Secondary DMA Controller's *Command* register enable bit and the mask bits for channels 0-3 and channel 4. They must be tracked from system initialization to guarantee that both the Remap Engine and the Distributed PCI Channels remain coherent.

Any write access to legacy DMA addresses 00D4h, 00DAh, or 00DFh which result in channel 4 being masked require the following modification to the Remap Engine algorithm:

1. The standard update to the Primary DMA channels 5, 6, and 7 takes place as usual with the channel 4 tracking bit also being updated. In addition, new spawned cycles must set the mask bits of channels 0-3 to reflect channel 4 being masked. Note that the original mask status for channels 0-3 is not lost, as the Remap Engine has been recording their status in a tracking bit.
2. New accesses to the Secondary DMA channels 0-3 that affect their mask status no longer result in spawned cycles. Instead, the Remap Engine just updates its mask tracking bits. As long as channel 4 remains masked, the corresponding bits in the Distributed PCI Channels will reflect this configuration.
3. If a write access to legacy DMA registers 00D4h, 00DAh or 00DFh results in channel 4 being unmasked, the Primary DMA channels 5, 6, and 7 are again updated as usual. This time, the Remap Engine must update the Secondary DMA channels (0, 1, 2, and 3) with its tracking bit values. Note that just because channel 4 is unmasked does not automatically result in channels 0, 1, 2 and 3 being unmasked; instead, their recorded individual states are restored.
4. Only when channel 4 is unmasked does the system return to a "standard" configuration mode where the individual mask settings for channels 0-3 control their channels' mask status.

Similar to the effect of channel 4 on the mask bits is the effect of channel 4 being disabled. For example, when channel 4 is disabled, Secondary DMA channels 0, 1, 2, and 3 must also be disabled. When channel 4 is re-enabled, the Remap Engine uses the tracking bit for the enable status of Secondary DMA channels 0-3 to update the enable bits for Secondary DMA channels 0, 1, 2, and 3.

### **Upgrading Existing PCI Systems to Support Distributed DMA**

Field upgrades for Distributed DMA require the systems to provide at least one primary PCI bus slot. In addition, the legacy expansion bus must be connected to PCI via a subtractive decode agent. Specifically, the legacy DMA registers must be decoded using the PCI's subtractive decoding techniques. From this standard PCI slot a Distributed PCI Channel and Remap Engine upgrades can be retrofitted to any existing PCI system.

In such a system the Distributed PCI Channel interface is identical to that described above, however, the Remap Engine's algorithm must to be modified to accommodate the legacy DMA Controller on the expansion bus. The Remap Engine intercepts all legacy DMA Controller's register accesses, as defined earlier, claiming these cycles before the subtractive agent, and spawning cycles to the Distributed PCI channels. After the Distributed PCI Channels are serviced, the Remap Engine reissues the original host access to the legacy DMA Controller's register onto the PCI bus. This time the expansion bus will accept the cycle as a subtractive agent and update the 8237s accordingly. The Remap Engine must guarantee that any reissued commands never unmask or activate any legacy DMA channels that were claimed by a Distributed PCI Channel. After the reissued cycle is complete the Remap Engine waits for the requesting PCI agent to return thereby completing the delay transaction cycle.

### **The Performance Issue**

Table 4 compares the theoretical performance of Distributed DMA versus legacy DMA on ISA. Distributed DMA significantly improves the overall system performance while maintaining full PC compatibility with the legacy DMA Controller.



**Table 4 -- ISA vs. Distributed DMA Performance**

Transfer Type	ISA DMA device (Mbytes/sec)	Distributed DMA (Mbytes/sec)
8-bit	0.8	8
16-bit	1.6	15
16-bit type F	4	30
32-bit	Not available	30
32-bit burst	Not available	100-132

However, Distributed DMA carries a performance cost due to mapping and expanding the legacy DMA I/O accesses to multiple Distributed PCI Channels. The greatest performance cost occurs when the share registers are mapped. The shared registers include the *Command*, *Status*, *Master Clear*, *Multi-Channel Mask* and *Clear Mask* registers. All these registers result in four PCI cycles being spawned for channels 0-3, and three PCI cycles being spawned for channels 5-7. However, the impact of this 1-to-4 cycle expansion on the overall system performance is not significant for several reasons:

1. The number of programming cycles are negligible when compared to the number of data-transfer cycles.
2. The majority of the remapped cycles for Distributed DMA result in one spawned cycle.
3. The Distributed PCI Channel Initiator Logic can include data buffers, and burst their PCI cycles to memory to further improve performance.
4. In the future, new PCI drivers can be written, bypassing the Remap Engine, thereby removing the spawned cycles that result.

## CONCLUSION

As expansion bus devices move away from legacy buses to the PCI bus, the need to maintain the legacy buses slowly erodes but the need to maintain the legacy DMA programming model remains. *Distributed DMA Support for PCI Systems* is a solution that the PC industry is implementing today. This standard will aid in a smooth migration path for all legacy devices onto PCI.

## BIOGRAPHY

Dwight Riley is the primary liaison for Distributed DMA at Compaq. His work experience at Compaq has encompassed the technology groups in the Portable PC Division and the Systems Division. Prior to joining Compaq, he worked for IBM's Commercial Desktop Technology group and was also involved with the early OS/2 Development efforts in Boca Raton, FL.

## REFERENCES

Compaq Computer Corporation. "DMA Operations", *Technical Reference Guide, Extended Industry Standard Architecture Expansion Bus*, Second Edition, August 1990.

Evoy, David. "Distributed-DMA Techniques Allow Easy Migration from the ISA bus to the PCI bus." *EDN*, November 23, 1995.

Cirrus Logic, Compaq Computer Corporation, National Semiconductor, OPTi, Standard Microsystems Corporation, Texas Instruments, and VLSI Technology. *Distributed DMA Support for PCI Systems*. Available from <http://www.compaq.com>, December 1, 1995.

# **A PCI ACCELERATOR ARCHITECTURE FOR THE ADI SHARC DSP**



# **ALACRON, INC.**

- **I. Target Applications**
- **II. ADI 2106X ARCHITECTURE**
- **III. Alacron's PCI Accelerator**
- **IV. Summary**



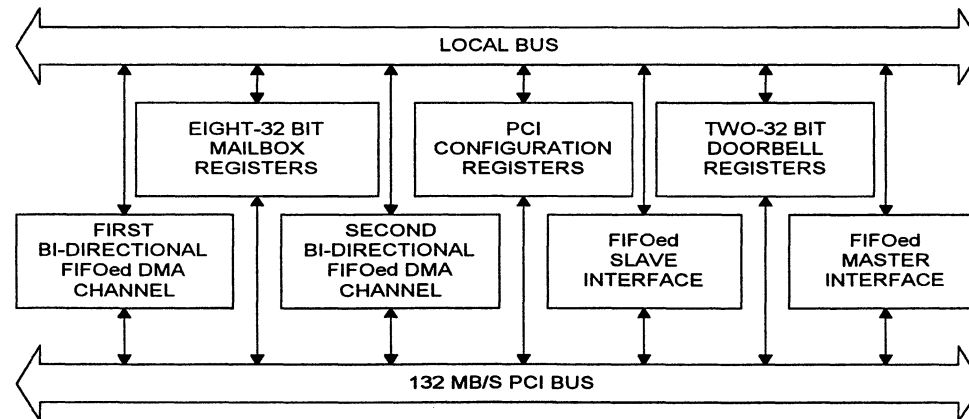
# Target Applications

- **Image Processing**
  - **Document Image Processing (OCR,ICR, Compression)**
  - **Medical Image Reconstruction**
  - **Biomedical Imaging**
  - **Pattern Recognition/Neural Networks**
  - **Automated Manufacturing**
- **Digital Signal Processing**
  - **Digital Receivers**
  - **Radar/Sonar**
  - **Commercial TeleComm**

# PCI Bus Architecture

- **Peripheral Component Interconnect Bus(PCI)**
  - **132/264 MB/s**
  - **33 MHz at 32 and 64 Bits Wide**
  - **Industry Wide Acceptance**
    - **Alpha, Power PC, Pentium**
  - **Heterogeneous Environment**
    - **PCI / ISA**
    - **PCI / EISA**
    - **PCI / NuBus**

# PCI BUS INTERFACE

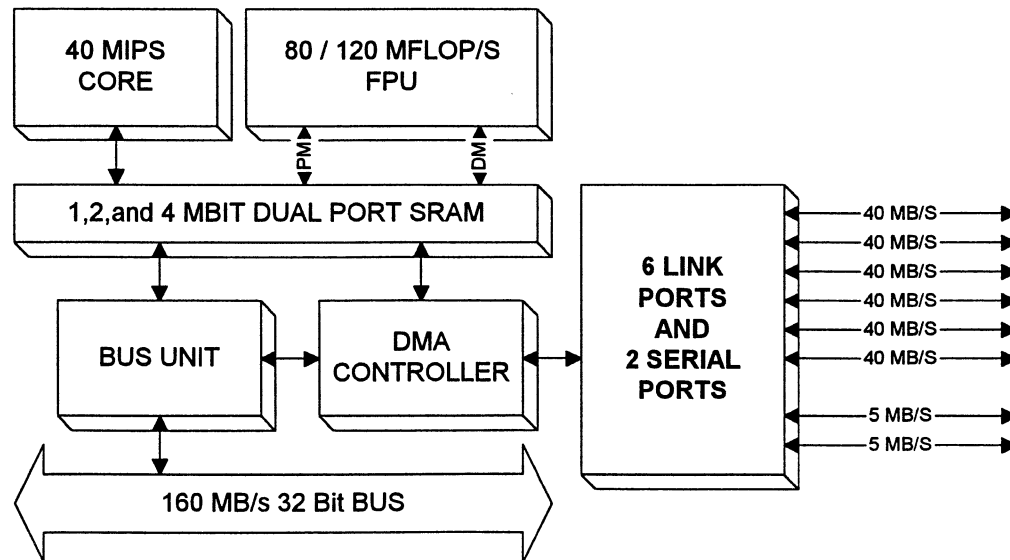


- **FIFO'D MASTER, SLAVE, AND DMA INTERFACE**
- **MAXIMUM BURSTS AT 132 MB/SEC**
- **BLOCK TRANSFERS AT 70 MB/SEC**
- **RANDOM READ/WRITES AT 8-12 MB/SEC**

# THE ADI 2106x ARCHITECTURE

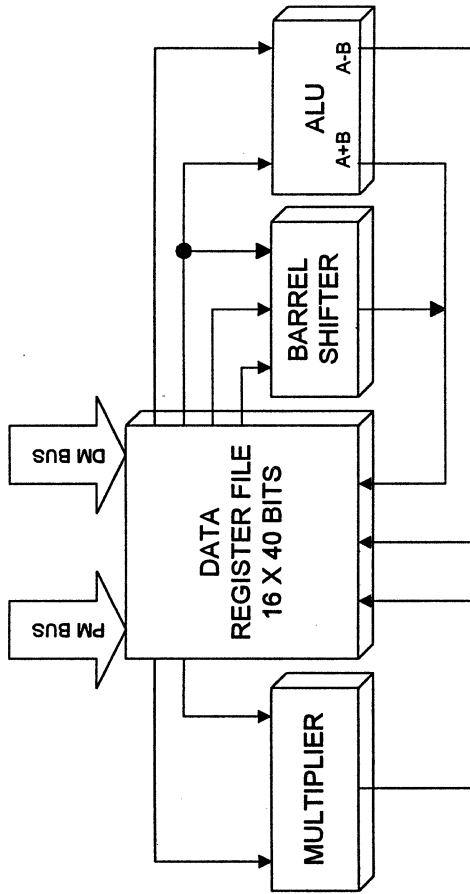


# The ADSP-2106x





# ADSP-2106X FPU



# ADI-2106X 'SHARC' SPECIFICATIONS

- 40 MIPS
- 120 MFLOPS (A+B,A-B)
- 80 MFLOP (OTHERWISE)
- 48 / 32 BIT EXTERNAL DATA BUS
- 1,2, OR 4 MBIT ON CHIP SRAM (21061, 21062, 21060)
- DMA CONTROLLER
- 2 SERIAL PORT (40 MBIT/S)
- 6 LINK POINTS (40 MBYTE/S)
- SIMD AND MIMD OPERATION
- DUAL PORTED SRAM
- CONCURRENT DMA WITH COMPUTATION

# PROCESSOR COMPARISON

Processor Attributes	ADSP-21060	TMS320C40	i860
	40 MHZ	40 MHZ	50 MHZ
Instruction Execution Time	25 ns	40 ns	20 ns
MIPS	40	25	50
Peak MFLOPS	120	50	100
Latency of Floating Point Inst.	25 ns	40 ns	60 ns
Internal RAM (32-Bit Words)	128K	2K	8K
On-Chip Memory Bandwidth	640 MB/S	N/A	320 MB/S
<u>I/O Capability</u>			
Serial Ports	2	0	0
Link Ports	6	6	0
DMA Channels	10	6	0
<u>DMA Bandwidth (MB/S)</u>			
Link Ports/Port	40 MB/s	32 MB/s	N/A
DMA Bandwidth	240 MB/s	50 MB/s	0
FFT, 1K Complex	460 usec	1,540 usec	520 usec
Divide, 32-Bit Floating Point	150 ns 6 cycles	360 ns 9 cycles	200 ns 10 cycles

# Alacron's ADI 2106x Accelerator Design



# **ALACRON'S SHARC DESIGN GOAL**

**OPTIMIZE PROCESSOR AND MEMORY  
PERFORMANCE FOR MIMD AND SIMD  
PROGRAM MODELS**

# MIMD programming model

- **Each SHARC Processor**
  - operates as an individual coprocessor
  - has separate thread of execution
  - may run entirely different code
  - 2106X application can implement inter-processor communication or cooperation if desired

# **SIMD programming model**

- **All 2106X processors operate in lock-step**
  - **executing same instructions simultaneously**
  - **process using different data.**
  - **easy to write for algorithms that may be partitioned in the spatial domain**
  - **no overhead for inter-processor synchronization when compared to MIMD parallel algorithms**



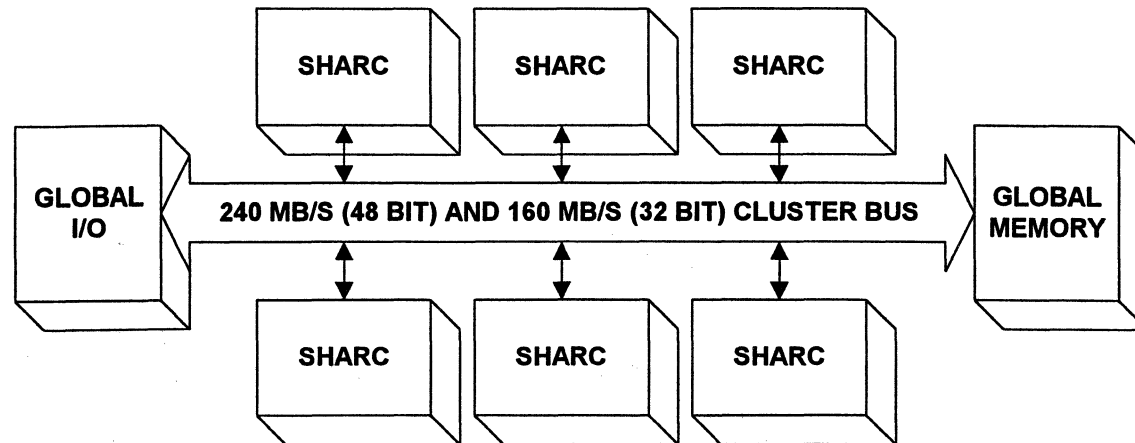
# SHARC DESIGN ALTERNATIVES

- STANDARD CLUSTER
- STANDARD MESH
- ALACRON DESIGN





# CLUSTER ARCHITECTURE



# CLUSTER ARCHITECTURE

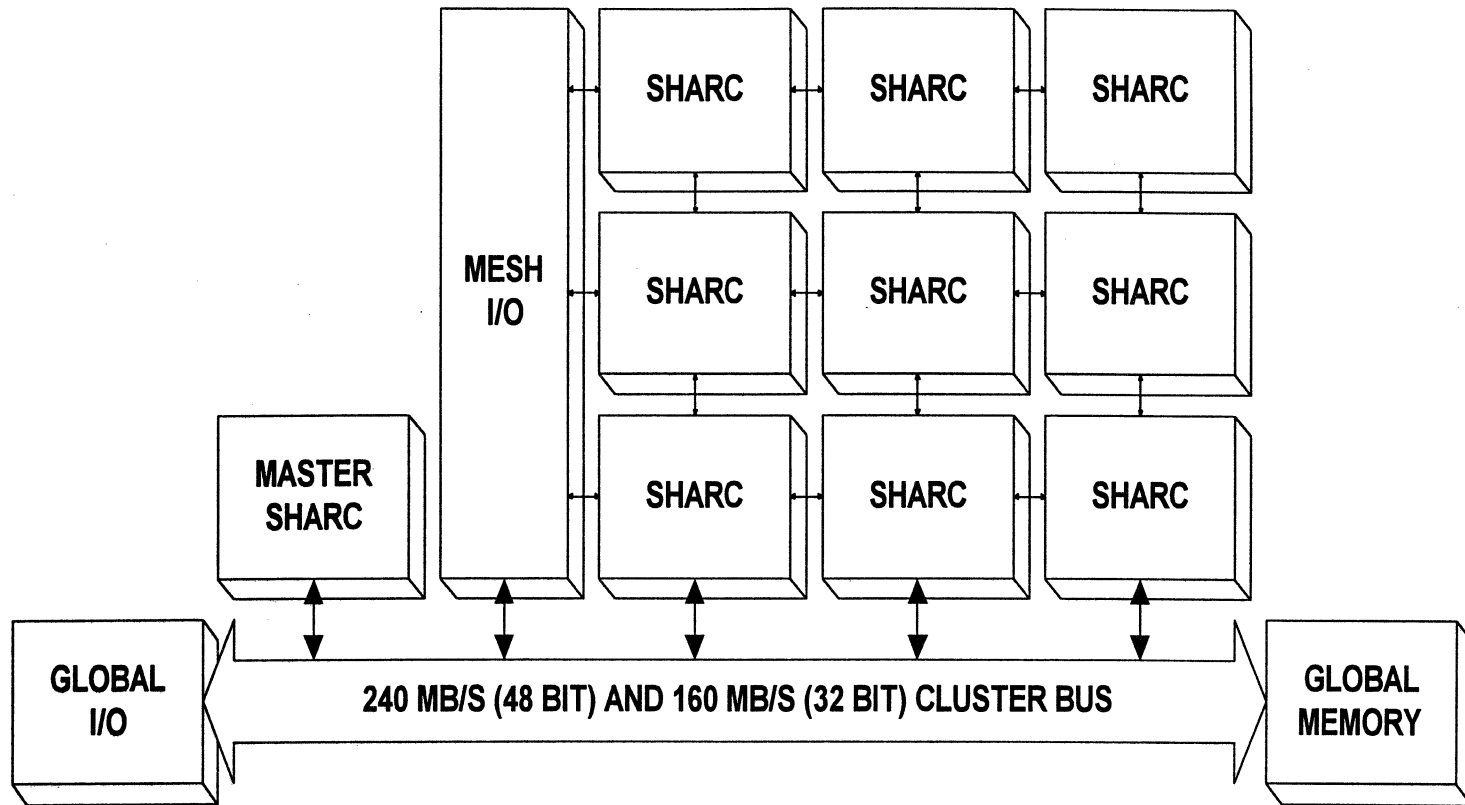
## Pros

- Easy to Program
- MIMD Operation

## Cons

- Processor Contention for Global Memory and I/O
- No SIMD Operation
- High Speed Memory Required
- Usually Implemented with Static Link Port Connections

# MESH ARCHITECTURE



# MESH ARCHITECTURE

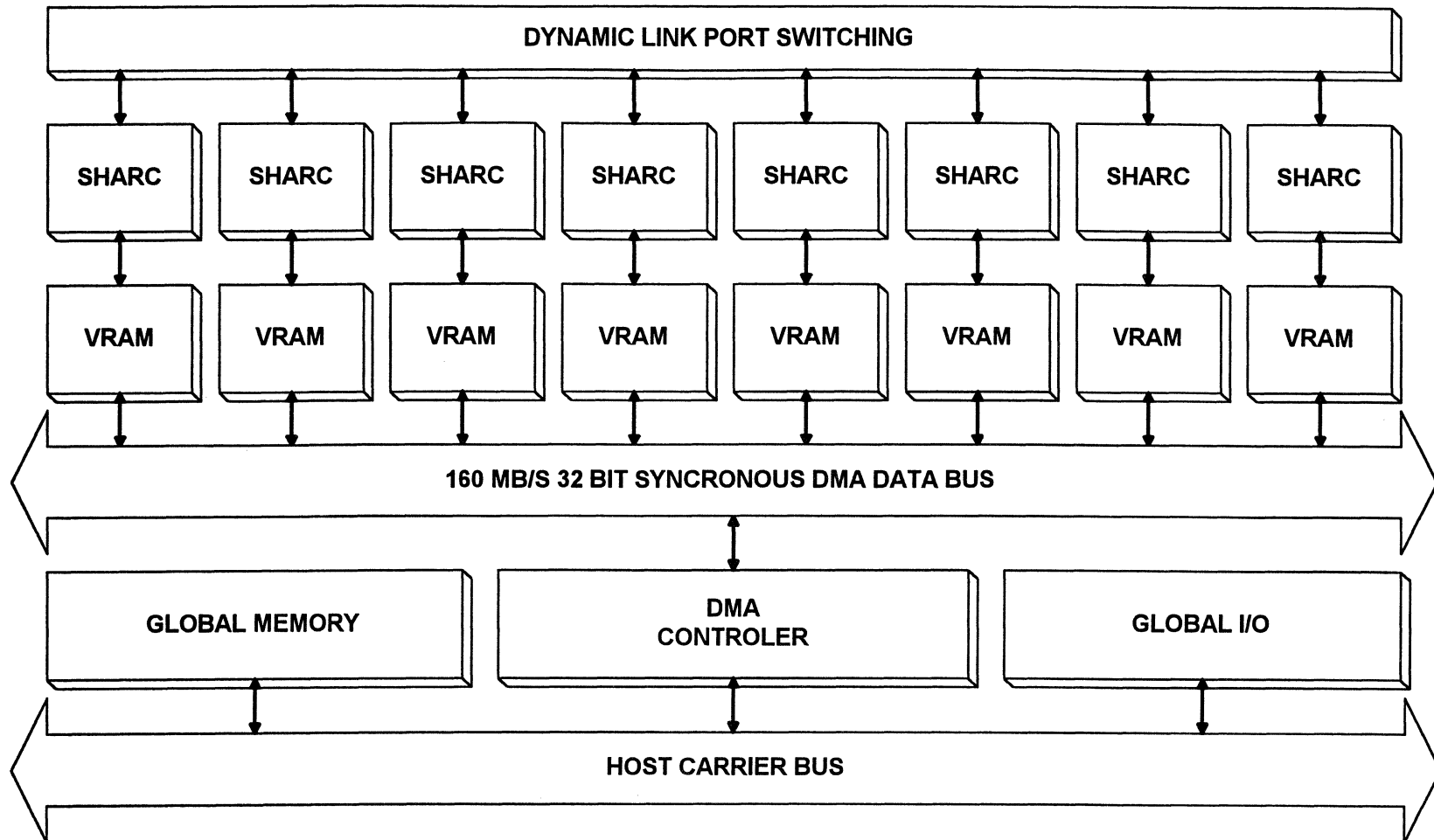
## Pros

- **Scaleable over a wide range of Applications**
- **Simple to understand and Program**
- **Single Copy of Instructions Required**

## Cons

- **Limited I/O Bandwidth**
- **SIMD Operation Only**
- **Fixed Link Port Topology**
- **Microcoding Required for Peak Performance**
- **Single Bus Passes All Instructions and Data**

# ALACRON ARCHITECTURE



138



# ALACRON ARCHITECTURE

## Pros

- SISD, MISD, SIMD and MIMD operation support (all modes)
- A continuation of the dual port memory concept in the SHARCs
- Processors do not contend for data on a common bus
- Link ports are configurable dynamically as required by the algorithms
- Dual port memory augments the SRAM internal to the SHARC chips
- Data and instruction duplication supported by DMA controller broadcast
- Data flows through the system without bottle-necks or bus contention
- Scalable across many applications

## Cons

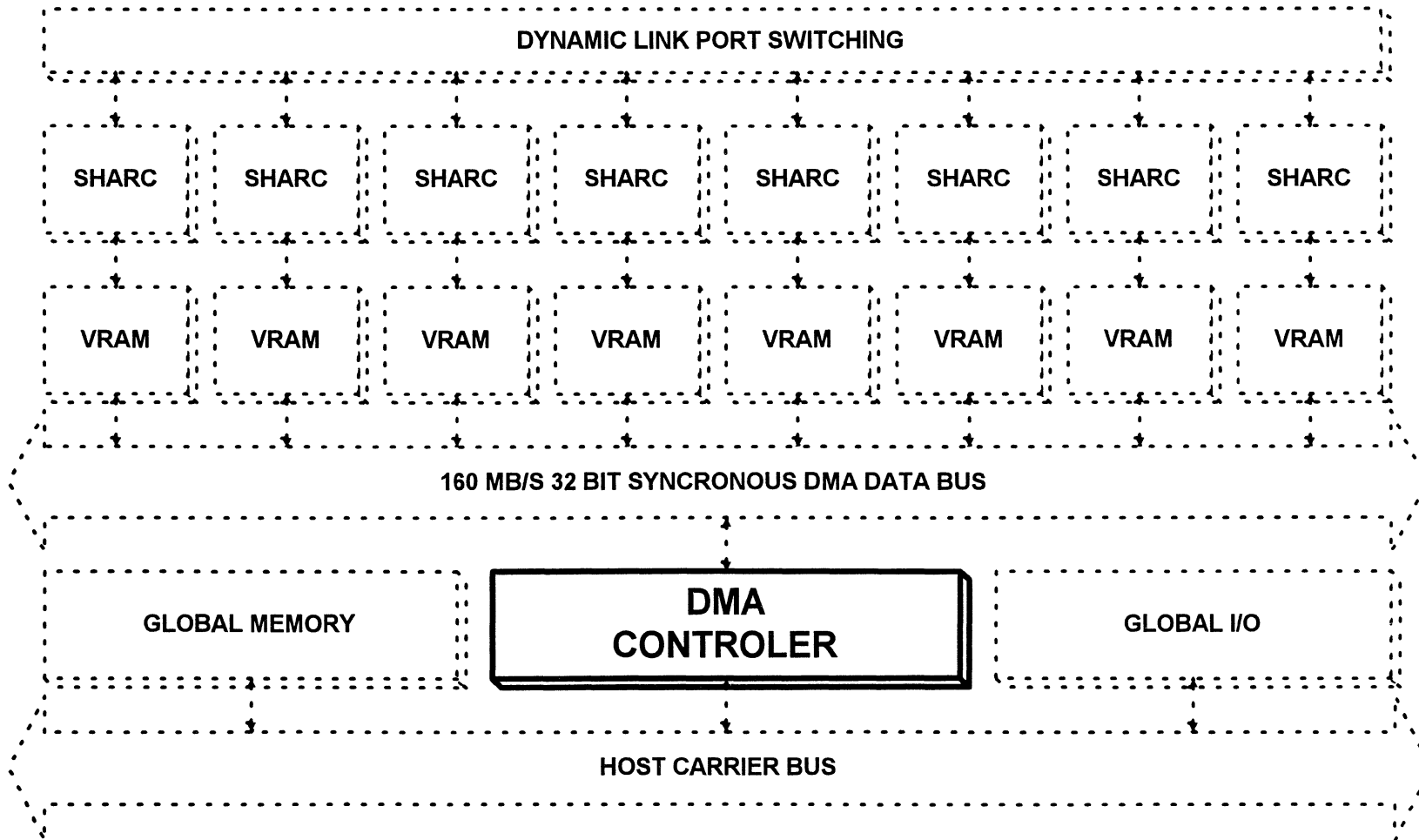
- Processors can not directly pass data over a common parallel bus
- SISD and MISD operation requires duplication of data
- SISD and SIMD operation requires duplication of instructions
- Microcoding required for peak performance

# COMPETITIVE PERFORMANCE COMPARISON

	Bus	Architecture	Memory Architecture	Link Port Config	Link Port Topology	SIMD	MIMD	Bus BW per SHARC	No Overhead Streaming Data Flow
ALACRON	PCI, ISA, or VME	Local Memory	1 MB Private VRAM 256 MB Global DRAM	Dynamic Universal	Direct Number Theoretic	Yes	Yes	80 MB/s	Yes
VENDOR 1	PCI	Mesh	Global SRAM	Static	Rectangular Mesh	Yes	No	Not Applicable	Yes
VENDOR 2	Proprietary	Cluster	Cluster Shared 512 KB FLASH and SRAM	Static	Star Cluster to Cluster	No	Yes	27 MB/s	No
VENDOR 3	VME	Cluster	3x2 Cross Bar 256 K SRAM 128 MB DRAM	Static	Star Adjustable by Cabling	No	Yes	20 MB/s	No
VENDOR 4	VME	Cluster	Cluster Shared 3 MB SRAM 512 K FLASH	Static	Adjustable by Cabling	No	Yes	40 MB/s	No



# ALACRON ARCHITECTURE

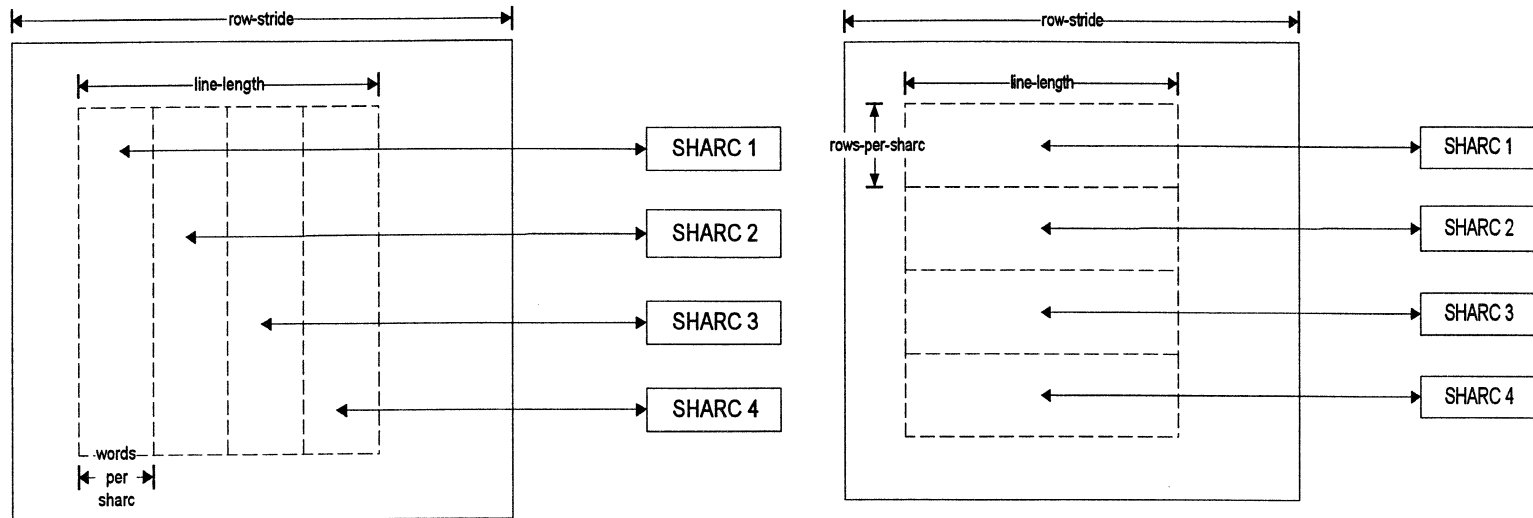


141



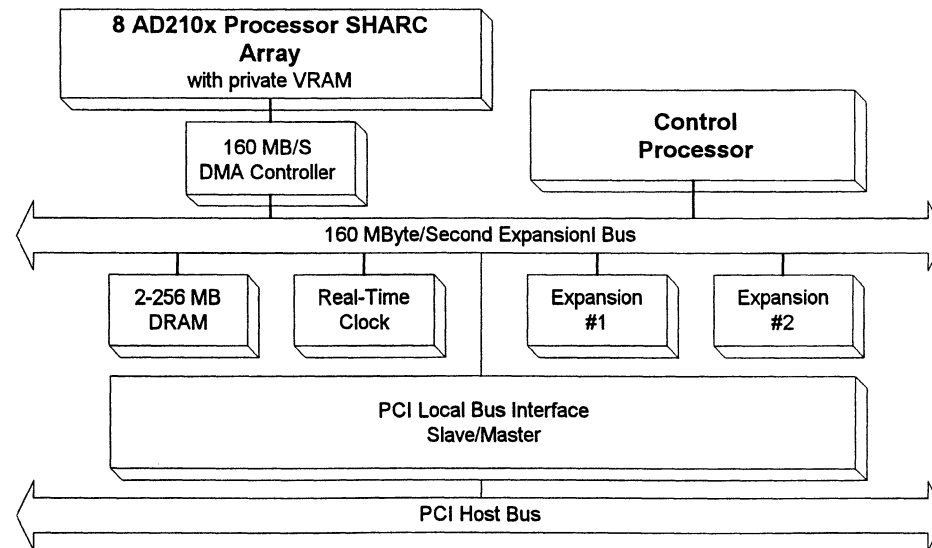


# TWO DIMENSIONAL DMA



- THE DMA ENGINE ALLOWS STRIDED TRANSFERS FROM MEMORY AND ALLOWS ON-THE-FLY TRANSPOSES

# SHARC MAIN BOARD DESIGN

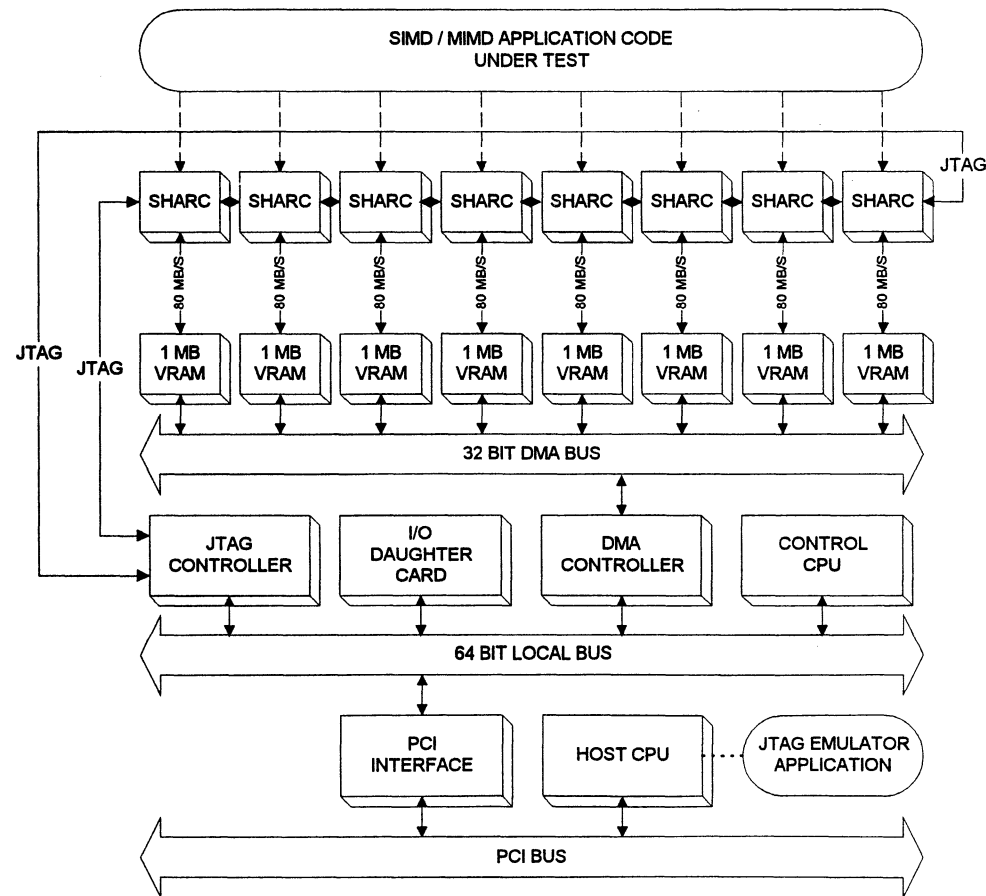


# SHARC BOARD FEATURES

- **Array of up to 8/16/24 2106xs (1-3 SHARC Modules)**
- **960/1920/2880 MFLOPS per Slot**
- **Up to 256 MB of DRAM SIMM Memory.**
- **Supports all Daughter Cards**
- **Two Standard Daughter Card Connectors**
- **Slave, Master, and DMA Interface**
- **PCI Interface**



# JTAG DEBUGGER ARCHITECTURE

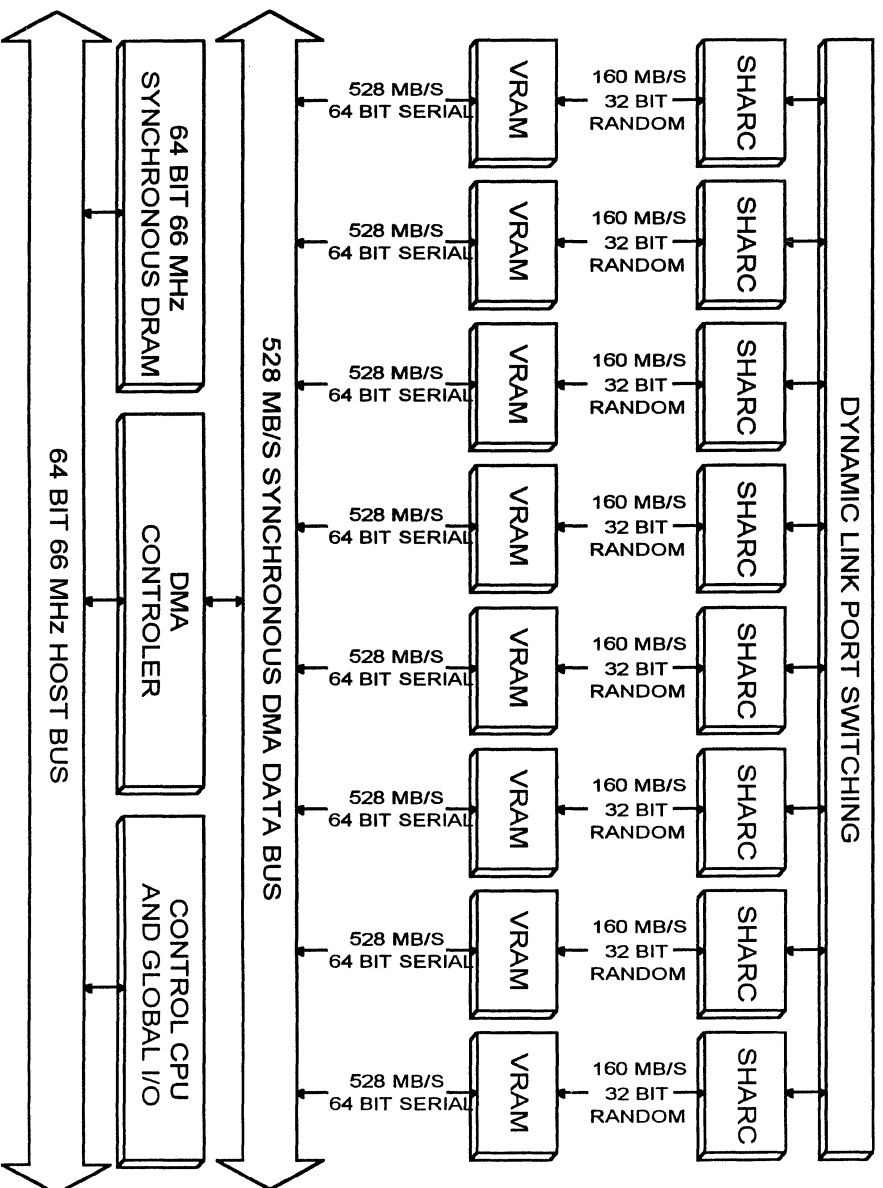


## **UTILITY OF JTAG BOUNDARY SCAN**

- **DESIGN FOR MANUFACTURE**
- **EFFICIENT GUI BASED HARDWARE  
DEBUGGER ENVIRONMENT**



# FUTURE ASIC BASED ENHANCEMENTS



# FUTURE ASIC BASED ENHANCEMENTS

- **Zero Wait State (160 MB/s) SHARC External Memory Accesses**
- **528 MB/s Peak DMA Bus Transfer Rates**
- **Less Than 500 Microseconds 512x512x8 Image Transfer Time**
- **Support For Both Video RAM and Graphic RAM**

# ARCHITECTURE COMPARISON

ARCHITECTURE	NUMBER OF CPUs	TOTAL MFLOPS	MFLOP/S PER SQ-IN STD-SMT	MFLOP/S PER SQ-IN BGA	MEMORY BANDWIDTH PER CPU	MEMORY BANDWIDTH PER CPU DURING DMA
i860 SHARED MEMORY	2	200	4	N/A	200	100
SHARC PRIVATE VRAM	8	960	19	28	80	80
SHARC PRIVATE VRAM	16	1920	24	39	160	160
SHARC PRIVATE VRAM	32	3840	29	53	160	160



# ARCHITECTURE COMPARISON

ARCHITECTURE	NUMBER OF CPUs	RELATIVE PRICE PER CPU	SIZE STD-SMT (SQ-IN)	Size BGA (SQ-IN)	MEMORY (MB)	POWER 5V Design WATTS	POWER 3.3V Design WATTS	MFLOP/S PER WATT(5V)	MFLOP/S PER WATT(3.3V)
i860 SHARED MEMORY	2	1	54	N/A	16	37	N/A	6	N/A
SHARC PRIVATE VRAM	8	0.75	52	38	12	68	47	14	23
SHARC PRIVATE VRAM	12	0.53	65	43	18	80	53	18	27
SHARC PRIVATE VRAM	16	0.55	78	49	24	92	60	21	32
SHARC PRIVATE VRAM	32	0.52	132	73	48	140	92	27	41



# ALGORITHM PERFORMANCE COMPARISON

Architecture	1D CFFT (1K) (MSEC)	2D CFFT (1Kx1K) (MSEC)	CONV5x5 (512x512) (MSEC)
1 x i860XP-50	0.547	1420	100
2 x i860XP-50	0.32	816	100
1 SHARC	0.457	1045	205
2 SHARC	0.26	527	103
4 SHARC	0.13	272	52
8 SHARC	0.072	138	23.5
16 SHARC	0.046	74	11.8
32 SHARC	0.024	39	6.2

# FFT PERFORMANCE COMPARISON

Length	8-SHARC (MSEC)	16-SHARC (MSEC)	32-SHARC (MSEC)	64-SHARC (MSEC)	1-i860XP (MSEC)	2-i860-XP (MSEC)
1,024	0.072	0.046	0.033	0.027	0.547	0.320
2,048	0.148	0.084	0.052	0.036	1.300	0.825
4,096	0.276	0.148	0.084	0.052	3.900	2.989
8,192	0.620	0.320	0.170	0.095	9.500	7.121
16,384	1.220	0.620	0.320	0.170	20.400	15.800
32,768	2.764	1.392	0.706	0.363	45.700	36.400
65,536	5.508	2.764	1.392	0.706	96.800	78.800
131,072	12.370	6.194	3.106	1.562	213.600	177.600
262,144	24.720	12.370	6.195	3.108	447.100	381.900
524,288	54.920	27.470	13.745	6.883	976.700	845.200
1048576	109.780	54.900	27.460	13.740	2133.623	1870.550



# Alacron's SHARC Design

- **Scalable**
- **High Performance**
- **Allows MIMD/ SIMD Operation**
- **Concurrent Data Transfer with Calculation**

## PMC: THE PCI MEZZANINE CARD

Rodger H. Hosking  
Pentek, Inc.  
55 Walnut Street  
Norwood, NJ 07648  
201-767-3994 (fax)  
e-mail: rodger@pentek.com

### ***Abstract***

PMC solutions have the capacity to improve performance while reducing the cost of mezzanine I/O expansion cards. From its start in the personal computer area, the PCI bus is now coming to the VMEbus.

A standard mezzanine card design, PMC is based on the Peripheral Component Interconnect (PCI) standard bus, which has been universally adopted for use as a high-performance local bus in Pentium-based personal computers. With its further adoption as a mezzanine expansion bus for VMEbus, VME board makers will be able to leverage off the economies of scale accruing to the personal computer industry.

This development also opens the floodgate of compatibility with other types of computers such as workstations, industrial and commercial computers, and other standard buses such as Multibus. This standard mezzanine bus has the potential to penetrate market areas which had previously remained proprietary, closely guarded or with high barriers to entry.

### ***Mezzanine Cards***

Mezzanine cards can satisfy three basic system design requirements:

- Provide a degree of flexibility to a host board such that a single host can be used in a variety of applications;
- Make it possible to stuff more components into a board's limited space;
- Add functions or enhancements to a board to extend product life.

Through the years, such cards have gone in and out of favor. Early add-on boards used what is by today's standards crude connector technology that was frequently prone to failure. In addition, there was often no mechanical support for the daughter boards other than the connectors.

But even as connectors improved, it was frequently considered a design goal to develop a board without add-ons. Boards with mezzanine expansion cards were looked upon as having design flaws and questionable reliability.

Within the last four or five years though, there's been an almost universal change of thought about mezzanine board technology. Even the most adamant of the holdouts, the U.S. military, has grudgingly acknowledged the benefit of such approaches with a number of factors contributing to the turnaround in thought. And, if any holdouts remain, PCI/PMC is expected to make believers of everyone.

### ***Previous Mezzanine Buses***

One of the first steps toward bringing mezzanine cards to some level of respectability was their adoption by Intel and other Multibus board manufacturers, who accepted the simple I/O concept of iSBX. This standard bus provided 8-bit I/O with limited bandwidth.

A short time later, Intel introduced its higher-performance iLBX which could serve as a local bus for memory expansion. This bus was migrated to Multibus II in the mid-80's and remains in use in many Multibus systems today.

About the time of Multibus II introduction, VME board manufacturers developed their own proprietary buses because of incompatibilities between the Intel and Motorola processors.

Still later, Intel developed its MIX bus for Multibus II. The MIX bus is in wide use in the Multibus community and has been successfully used by Pentek as an expansion bus for VME boards.

### ***The PCI Bus***

PCI is a local bus that interfaces with the processor and memory bus on one side, while it provides a high-speed channel on the peripheral expansion side. Such a bus solves a variety of problems:

- It provides local connection for other buses, such as ISA, EISA, or VMEbus;
- It makes available simple means to implement I/O expansion;
- It eliminates the need for motherboard redesign with each processor revision.

The PCI specification is inherently high performance allowing transfer rates of 132 Mbytes/sec in its 32-bit implementation. Options using the 64-bit version double that transfer rate. This kind of bandwidth brings PCI into the domain of very-high resolution graphics moving into the full motion video area. In addition, it lends itself to the new breed of high speed I/O such as Fiber Channel, ATM and FDDI.

The PMC specification, now known as IEEE P1386, defines the mechanical and electrical properties of the bus and the card. The physical size of the expansion card is roughly 3 x 5 inches, so it will fit comfortably on a 3U VME board. Two of them will fit on a 6U VME or Multibus board, and four of them will fit on a Futurebus+ card. In addition, the height of the board and connectors are specified so that a PMC will fit in a single slot board, such as shown in Figure 1.

### ***PMC I/O***

I/O for the PMC is brought out the backplane on the P2 connector. In addition, the specification allows for direct connection to the front panel of the VME board.

A separate PMC front panel can protrude flush with the VME front panel through the knockout as shown in Figure 1 and Figure 2. The pin connections have been specified to maximize signal integrity while assuring power distribution. For example, signal pins are guarded by ground or supply pins.

### ***PMC Benefits***

As an IEEE standard, PMC assures users that any host or module complying with the standard will function in any module or host that has been designed to the specifications. While this gives users the flexibility to mix-and-match different host cards with different option modules, it also gives vendors the ability to design basic host boards without special consideration to interface I/O. The fact that PMC is an open standard allows OEM's with nonstandard buses to take advantage of the same leverage as makers of standard buses.

The second advantage of using PMC is that it provides a large measure of stability. PMC provides a standard, high-performance local bus that will remain the same from processor to processor. Only the processor-to-memory bus need be modified.

Performance, of course, is another key element of PMC. New graphics and GUI's, extensive use of imaging, video and faster communications have placed a major demand on processor, I/O, and system bandwidth. PMC will go a long way to alleviate the I/O bottleneck. With a bandwidth of 132 Mbytes/sec for a 32-bit implementation and 264 Mbytes/sec for a 64-bit version, PMC is capable of handling just about everything up through ATM and full-motion video.

PCI and PMC put the focus on the main objective of the standard-bus community trying to provide a standard, off-the-shelf alternative to costly proprietary design. PMC will go a long way in providing that capability with a broad range of standard I/O.

VME will continue to drag along a large number of mezzanine cards with special functions. Some will be low-performance 8-bit I/O such as IndustryPack, others may be part of multiprocessing configurations. But there is little question that every system will include at least one or two PMC modules in the very near future.

### ***Pentek PMC Offerings***

Pentek is introducing a complete line of PMC modules which provide functions that utilize the same areas of expertise developed in our MIX module family. These include DSP coprocessors, T1/E1 telecom interfaces, digital receivers and data acquisition functions.

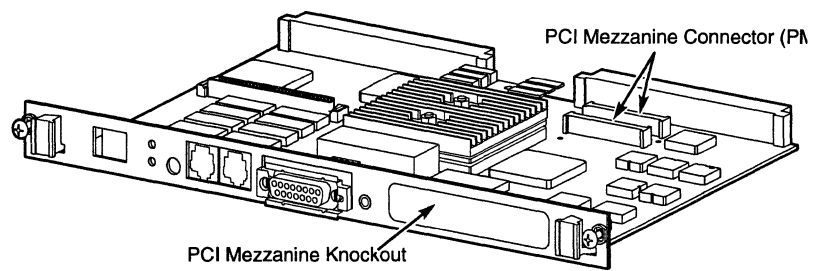
The Model 7110 'C44 DSP coprocessor shown in Figure 3 is our first coprocessor. Pentek is also introducing PMC baseboards capable of accepting one or two PMC modules. Shown in Figure 4, the Model 4285 Octal 'C40 VME board is our first PMC baseboard offering.

As new devices for DSP peripherals become available, they will be incorporated in Pentek's product line in both MIX and PMC formats.

### ***References***

For more information on the PMC/PCI bus, refer to:

1. Digital Equipment Corporation White Paper: *PCI PMC: A Local Bus with Global Importance*, September 1994
2. Dick Somes and Wayne Adams, Digital Equipment Corporation: *A Case for the PCI Mezzanine Card Standard*, I&CS October 1995
3. IEEE P1386.1: *PCI Mezzanine Card*, IEEE Standards



"Alpha AXP VMEbus SBC with PCI Mezzanine Connector"

Figure 1. 6U VME Board shows the PCI connectors and front panel I/O knockout before the PMC is installed. (Courtesy of Digital Equipment Corp.)

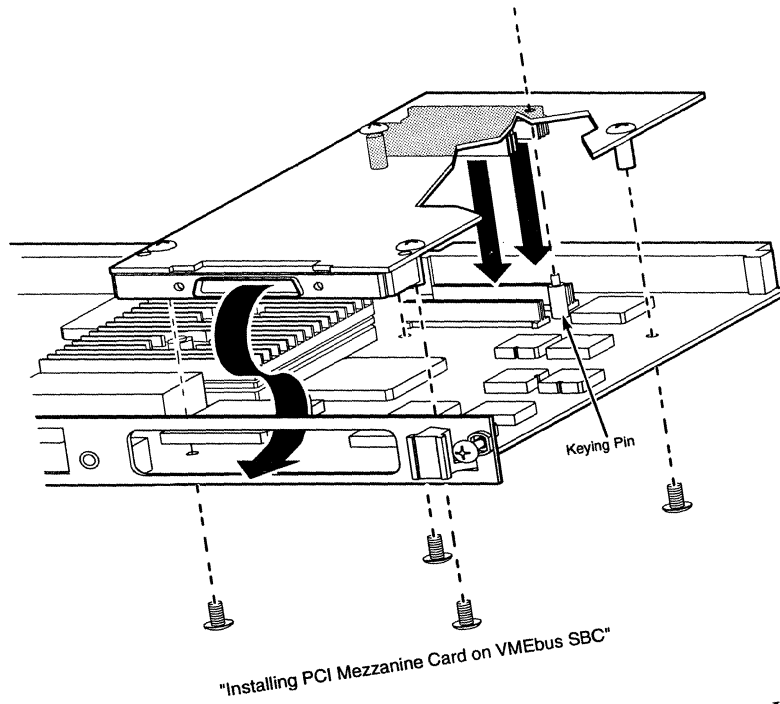


Figure 2. This figure illustrates how the PMC is installed on the VME board and shows the standoffs and mounting screws. Note how I/O can be taken directly out from the front panel through the PMC knockout. (Courtesy of Digital Equipment Corp.)



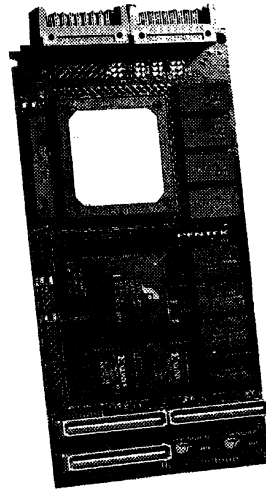


Figure 3. Pentek Model 7110 TMS320C44 PMC Module

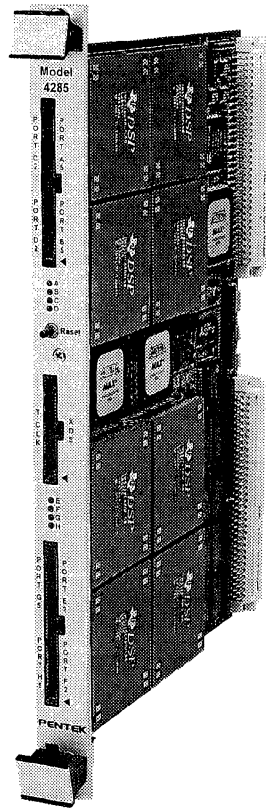


Figure 4. Pentek Model 4285 Octal  
TMS320C40 Processor PMC Baseboard

## **DSP AND I/O SYSTEM INTEGRATION FOR PCI**

Jack Carter and Manish Kasliwal

Sonitech International Inc.

14 Mica Lane

Wellesley, MA 02181

(617)235-6824/2531 (fax)

e-mail jack@sonitech.com

**The high speed PCI bus offers exciting integration possibilities for DSP systems requiring external I/O. Previous PC bus standards were incapable of providing the bandwidth needed for high performance real-time systems. Today, however, complex DSP based solutions can be implemented with dedicated boards residing on the PCI bus. This paper will address hardware and software issues related to integrating real-time DSP and I/O applications on PCI. Considerations for selecting DSP and I/O subsystems will be presented. System software issues will be examined. Finally, a sample application will be discussed including potential bottlenecks and pitfalls.**

## RACEWAY INTERLINK AS A PCI SWITCHING FABRIC

Barry Isenstein and Bob Blau  
Mercury Computer Systems, Inc.  
199 Riverneck Road  
Chelmsford, MA 01824 USA  
(508) 256-1300 FAX (508) 256-3599  
email: barry\_isenstein@mc.com bob\_blau@mc.com  
URL: <http://www.mc.com>

### **ABSTRACT**

This paper introduces a high-performance methodology for linking large numbers of PCI devices. Since a single PCI bus segment can support only 10 loads, many multiprocessing and switching applications require extended PCI connectivity. The methodology described conforms to the PCI 1.0 Bridge Specification, without imposing a hierarchical bus structure that limits scalability. Rather, a switching fabric is defined that facilitates multiple simultaneous PCI-to-PCI transactions. The switching fabric was designed specifically to provide maximum PCI bandwidth, low latency, and minimum contention.

A new bridging device called the PXB from Cypress Semiconductor and Mercury Computer Systems enables an existing switching fabric called RACEway [1, 2] to be used with PCI endpoints. The PXB-RACEway technology allows up to 256 PCI bus segments within a single system.

PXB and RACEway can be used in many different packaging scenarios, including PC motherboard designs and backplane paradigms. A case study used for illustration is the VMEbus packaging format. Two standards in this environment have solved the many mechanical and electrical issues required to develop robust, open solutions. The PCI Mezzanine Card (PMC) [8] standard allows the VME community to exploit PCI devices and processors. The RACEway Interlink standard provides the high-bandwidth, low-latency switched-fabric interconnect for VME and PMC. RACEway and PXB have no dependency on VME. Additional PCI switching applications are also being developed.

### **PC LIMITATIONS AND HIERARCHICAL BUS STRUCTURES**

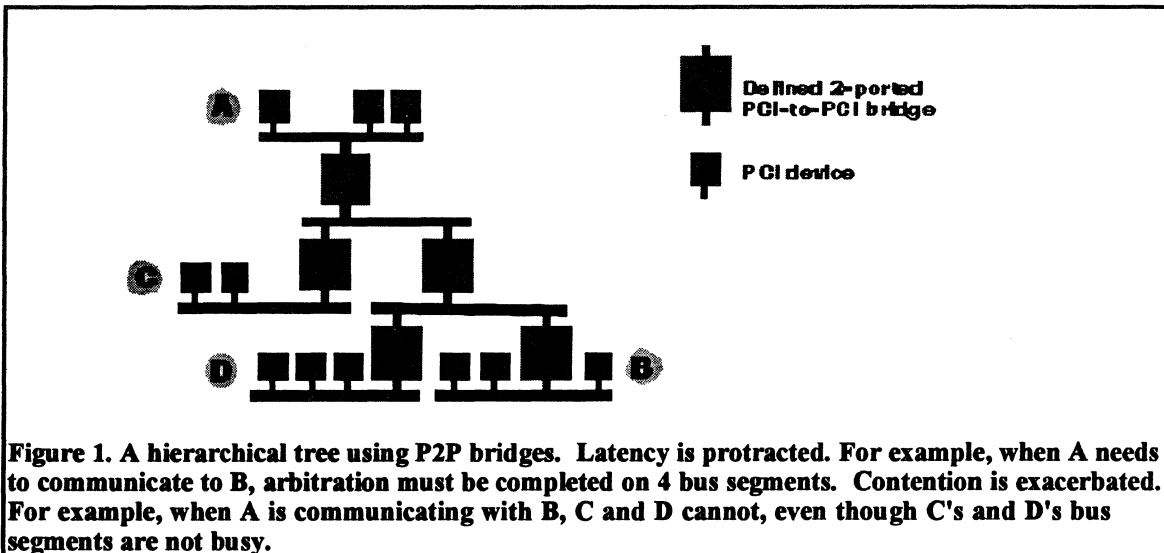
PCI has architectural and physical constraints which limit its ability to efficiently scale to support large numbers of high-speed processors and peripherals.

#### **Electrical Limitations**

On the physical side, PCI has electrical and mechanical constraints to ensure signal integrity. These include limits on loading, trace lengths, and connectors. These typically limit a single PCI bus segment to a total of 10 loads, with each connector considered as two loads. Thus, a motherboard or passive backplane typically has a limit of four plug-in boards on a single PCI bus segment. In the future, with 66MHz PCI, this limit will be two plug-in boards on a motherboard or a passive backplane due to a limit of only five loads per PCI bus segment. In five years time, the frequency requirements will mandate point-to-point connections, necessitating a switching fabric interconnect between PCI agents.

#### **Architectural Limitations**

To overcome the physical limitations of a single PCI bus segment, PCI-to-PCI bridges are used. Bridging allows connecting up to 256 PCI bus segments together. However, the bridge specification defines only dual ported PCI-to-PCI bus bridges (P2P), resulting in the need to connect the 256 bus segments as a hierarchical tree. This tree is an inefficient interconnect topology for larger systems with poor contention, latency, and bisection bandwidth characteristics.



**Figure 1. A hierarchical tree using P2P bridges. Latency is protracted. For example, when A needs to communicate to B, arbitration must be completed on 4 bus segments. Contention is exacerbated. For example, when A is communicating with B, C and D cannot, even though C's and D's bus segments are not busy.**

## **IMPROVING PCI CONNECTIVITY**

The solution to improved PCI connectivity is to connect PCI bus segments using a switching fabric instead of a bus hierarchy. A switching fabric provides point-to-point interconnects with the following features:

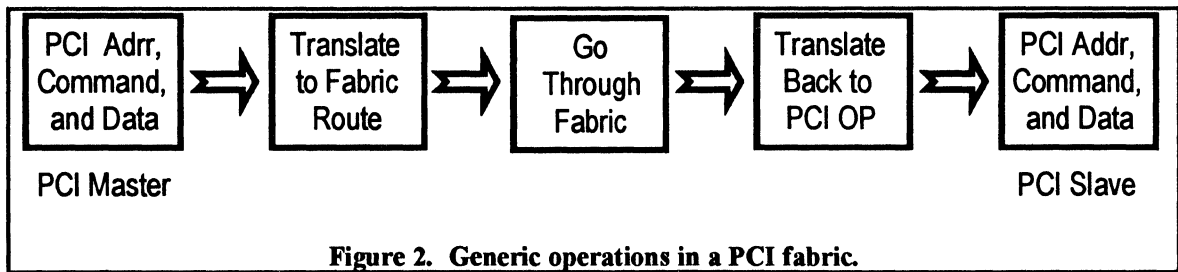
- high bandwidth
- low latency
- multiple simultaneous transactions
- scalable interconnect topologies
- real-time features desirable for I/O (priority-based preemption)

### **Switching Fabrics**

A switching fabric is an interconnection architecture which uses multiple stages of switches to route transactions between an initiator and a target. One benefit of switching fabrics is that each connection is a point-to-point link. This inherently provides better electrical characteristics allowing higher frequencies and greater throughput than bus architectures. The use of multi-stage switching also allows flexibility and scalability in the size and topology of the interconnect. Examples of prevalent switching-fabric standards are ATM at the WAN and LAN levels, and RACEway at the board and chassis levels.

Each stage of a switching fabric typically comprises an intelligent, multiport crossbar switch. The switch device recognizes a data-stream header message to dynamically route the interconnect transaction through the appropriate port to the next stage. The PCI to switching-fabric interface must forward transactions from one PCI bus segment to another. In the fabric interface, PCI addresses are translated to crossbar switch route and fabric addresses, and then back to the original addresses at the destination PCI bus, as shown in figure 2.

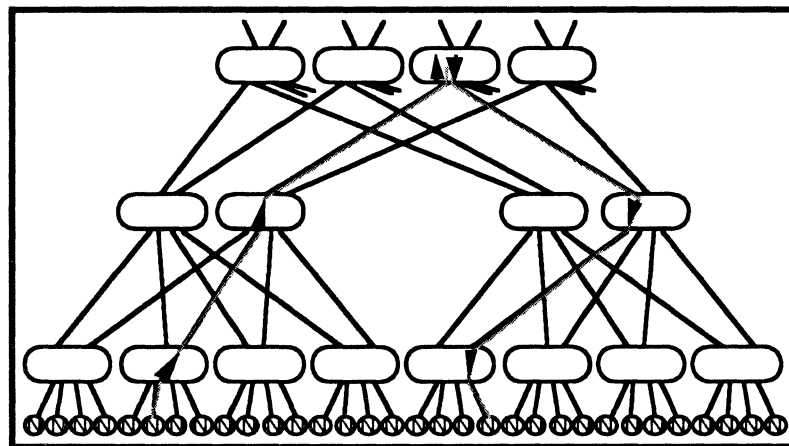
Switching fabrics typically provide redundant interconnection resources to allow multiple transactions to proceed at the same time. This improves aggregate throughput by reducing contention and latency, and can also provide improved fault resiliency.



### RACEway Switching Fabric

RACEway is a switching fabric for high-performance, real-time embedded applications. Detailed descriptions can be found elsewhere [1, 2, 3, 4, 5, 9]. RACEway interconnects are accomplished using a silicon building block called the RACEway Crossbar that occupies about one square inch of circuit-board real estate, and consumes about one watt of power. RACEway scales up to hundreds of nodes and provides for multiple Gbytes/second of throughput with deterministic latencies measured in microseconds. RACEway is commercially available as chips and modules from Cypress Semiconductor, and as board-level products and integrated systems from Mercury and other vendors.

The current RACEway Crossbar is a six-ported device with each port capable of bidirectional transfers at 160 Mbytes/s. Since each crossbar is fully connected, three simultaneous transfers can take place for an aggregate 480 Mbytes/s. RACEway Crossbars connect gluelessly (no additional circuitry) to form various topologies. A key advantage of RACEway is its topology independence; fat-trees (figure 3), meshes, rings, and pipelines have all been implemented using the same crossbar building block.



**Figure 3. A RACEway fat-tree. Each switch is a RACEway Crossbar. "N" stands for node which could represent a processor, an I/O device, or a bridge to a bus.**

RACEway employs a variant of circuit-switching but with additional features to remove the contention issues usually associated with circuit-switched networks. The circuit-switched nature of RACEway provides for read and read-modify-write operations that are very important for low latency bus bridges. The other features; pre-emption, adaptive routing, and split transactions provide for fabric contention control. RACEway also supports broadcast and multicast operations for applications that require multiple destinations per single transfer.

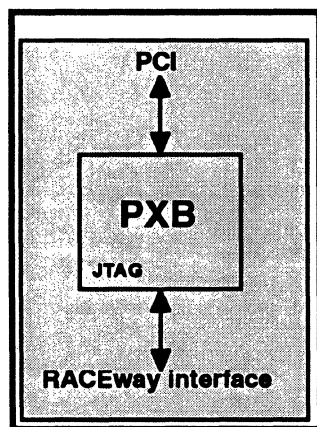
Arbitration in RACEway is fast, on the order of 125 nanoseconds per crossbar in a given path (collection of crossbar ports from master to slave). In the largest systems, typical latencies are under a few microseconds. RACEway features a priority pre-emption mechanism, so guaranteed worst-case latencies are still in the single digit microseconds. An adaptive routing mechanism provides a measure of transparent contention control for certain topologies by automatically selecting non-busy paths.

For slow devices, RACEway transactions can be split. For example, when reading a slow device the master requests a read over a RACEway path, releases the path, and waits for the slave to send the requested data. In this way, crossbar ports are free for other transactions while the slow slave gathers the data.

An example of a 32-node fat-tree topology is illustrated in figure 3. Fat-trees as large as 192 nodes have been implemented with larger systems planned. In figure 3, it is possible to have up to 16 simultaneous transfers for a total aggregate peak bandwidth of 1.28 Gbytes/s with a bisection bandwidth of 640 Mbytes/s.

## PXB

The PXB (figure 4) is a single-chip bridge between PCI and RACEway, allowing nodes on RACEway to be PCI buses. In referring to figure 1, the PXB performs the "translate" functions on either side of the RACEway fabric. One port is a standard 2.1-compliant PCI interface at 33 MHz. The other port is a standard 1.5.1-compliant RACEway interface. Maximum burst rates of 132 Mbytes/s are supported with an anticipated 100 Mbytes/s sustained performance on 256-byte blocks.



**Figure 4. The PXB is a single-chip (144-pin device) solution that bridges PCI and RACEway.**

Features supported by the PXB are:

- ANSI/VITA 5-1994 RACEway Specification compliant
- 2.1 PCI Local Bus Specification compliant (see below)
- 1.0 PCI to PCI Bridge Specification compliant
- 256 PCI segments supported
- 100 Mbytes/s sustained
- Memory, I/O, and configuration operations
- 32x32-bit write posting
- 32x32-bit read prefetching
- Coupled operations
- PCI arbitration
- 3V/5V I/O; 5V logic
- JTAG
- 144-pin PQFP

PCI options unsupported:

- 64-bit data path
- 66 MHz
- Interrupt Acknowledge command
- Special Cycle command

- Dual Address Cycle command
- Cache support
- Sideband signals

## PCI-RACEWAY FABRIC CHARACTERISTICS

There are several critical issues that any PCI fabric must address. This section outlines how the PXB with RACEway meet many of the difficult challenges facing PCI fabrics.

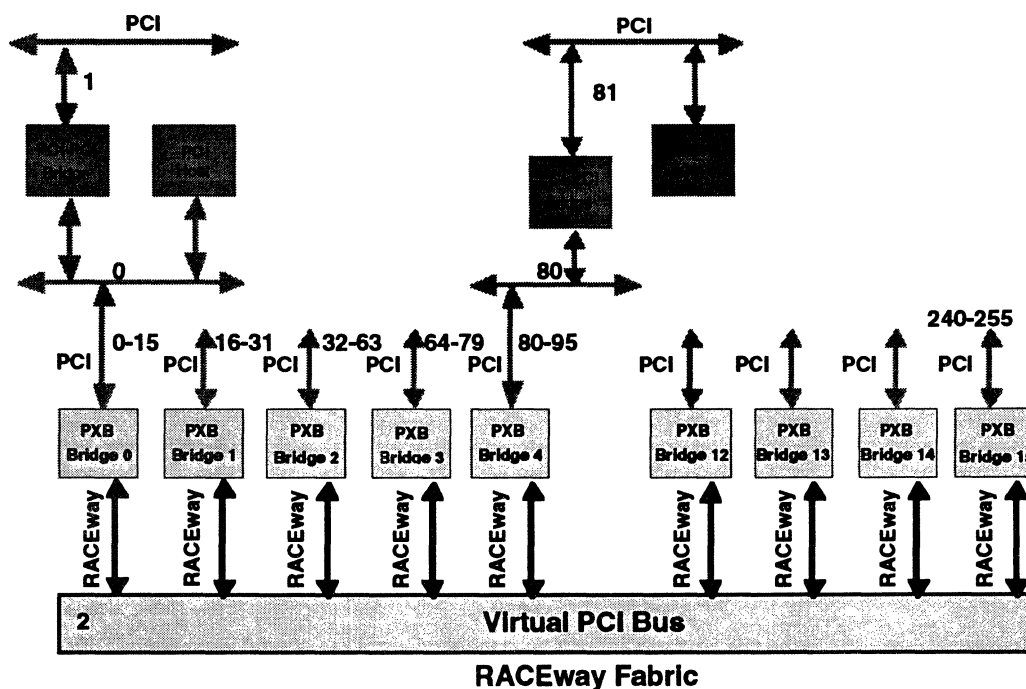
### Transparency

Bridge-specific knowledge is necessary during initialization to configure the bridges for transparent operation. The PXB specification is an extension of the existing PCI bridge specification, not a redefinition.

The underlying principles of the PXB are:

- Normal run-time software does not need to know that a RACEway fabric is present.
- The only software that needs knowledge about the RACEway switching fabric is the Power-On Self Test (POST) code in the BIOS and OS.
- The only software that needs knowledge of the internals of the switching fabric is loadable from an expansion ROM by the POST code.

During the POST execution, the PXB is initialized. In effect, RACEway becomes a virtual PCI bus segment (figure 5) with up to 16 bridges on that segment. (PCI configuration operations allow for a maximum of 32 devices or bridges on a PCI segment).



**Figure 5.** After completion of POST, the RACEway fabric "looks" like a PCI bus with up to 16 standard two-ported bridges. The numbers in the diagram represent PCI bus numbers that are assigned in the configuration phase of POST.



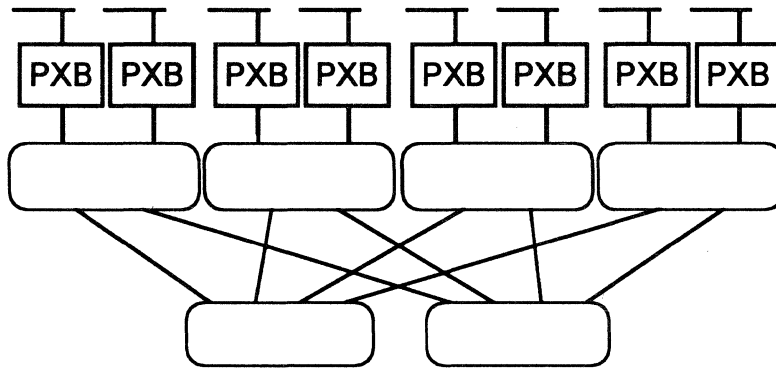
Unmodified driver and application software logically treat all PCI devices in this configuration as if they were bridged using the existing bridge specification. Physically these 16 PCI bridges represent up to 16 VMEbus slot. Each slot can represent up to 16 PCI bus segments, providing for a potential maximum of 256 PCI bus segments.

The PXB automatically performs the necessary translation as shown in figure 2. The PXB is viewed as a two-ported bridge but actually provides the high performance and low contention benefits of a switched fabric.

#### **Performance: Bandwidth**

Each RACEway point-to-point link is capable of 160 Mbytes/s, well above PCI's 132 Mbytes/s rates. A major advantage of a switching fabric over a hierarchical bridging topology is that aggregate and bisection bandwidth is improved by using point-to-point interconnects with multiple simultaneous transactions through the switching fabric.

For example, consider the eight-node RACEway configuration in figure 6 and a 100 Mbytes/s sustained rate for PCI/PXB. Each PCI bus can operate independently or use RACEway to communicate between buses. With four simultaneous RACEway transfers, the system in figure 6 has an aggregate interconnect bandwidth of 400 Mbytes/s and a bisection bandwidth of 400 Mbytes/s (a nonblocking configuration).



**Figure 6. A small scale PCI RACEway switching fabric.**

#### **Performance: Latency**

RACEway was designed for low-latency, real-time operation. To completely establish and initiate an internode transfer across RACEway, including all protocol overhead takes approximately 150 nanoseconds, plus 125 nanoseconds of latency per crossbar for the first data word and zero additional latency for each subsequent word in a block transfer. In figure 6, no more than three crossbar "hops" are necessary, resulting in a fabric interconnect latency of approximately 525 nanoseconds; total latency will be about one microsecond considering the PXBs and PCI latency.

The RACEway protocol allows for variable size data blocks up to 2048 bytes, offering amortization benefits for applications that use long transfers. To support this, the PXB supports combining sequential data phases into bursts, write posting, and read prefetching.

#### **Performance: Contention**

The RACEway protocol and the PXB support delayed operations (referred to as split transactions in RACEway documentation) which lessen fabric contention. For example, during a delayed read operation, the fabric is free for other transactions until the source of the read is ready to respond with data.

The RACEway protocol is priority-preemptable. Real-time applications with sensor data can choose to take advantage of priorities to minimize latency. Latency for the highest-priority data is deterministically guaranteed regardless of fabric traffic<sup>1</sup>.

### **PCI Reads and Locked Accesses**

RACEway is a preemptable circuit-switched fabric that supports split transactions. The circuit-switched features of RACEway allow the PXB to accomplish PCI reads and locked accesses. These operations and PCI coupled operations require that the source and destination nodes establish a logical circuit to guarantee PCI response times for completion.

However, as noted above, the RACEway PCI fabric also supports write posting, read prefetching, delayed operations, multiple concurrent transactions, and priority preemption. These features remove or alleviate the contention problems typically associated with conventional circuit-switched topologies.

### **Packaging and Power**

RACEway is very efficient in power and real estate consumption. The RACEway Crossbar is packaged in a ball grid array that is about one square-inch and consumes approximately one watt. The PXB chip is targeted to be less than one square-inch and about one watt.

Each RACEway port has 40 wires and uses conventional 40 MHz CMOS signaling. No exotic design or manufacturing techniques are required to overcome wire density or crosstalk problems. (RACEway signaling through VMEbus connectors, concurrent with VMEbus transactions, has been proven and field-tested in more than 500 systems to date.)

RACEway's modest footprint and electrical requirements make it suitable for both embedded and desktop platforms. The total interconnect and interfacing requirement in figure 6 is accomplished with six crossbar chips and eight PXBs consuming less than 14 square inches and approximately 14 watts. If one considers an eight-slot system configuration for figure 6, this averages to 1.75 square inches and 1.75 watts per slot. (In the VME packaging paradigm, we conservatively estimate that RACEway and PXB use less than two percent of the real estate budget and less than five percent of the power budget to execute the total interconnect and interfacing requirement.)

### **Strong Ordering and Coupled Operations**

The PXB supports PCI 2.1 coupled operations. Drivers using I/O commands can be ensured of strong ordering.

### **Interrupts**

Internally to the PXB and transparent to the PCI bus (and software), the PXB uses a mailbox scheme to handle interrupts through the fabric. A PCI interrupt is converted to a "message" at the source PXB that writes to a specific mailbox in the destination PXB. The destination PXB responds to that mailbox location by posting the appropriate PCI interrupt. Additional logic in the mailbox facility allows multiple devices on the secondary side of the switching fabric to share the same interrupt line.

### **RACEWAY INTERLINK**

RACEway Interlink is an ANSI standard [1] that specifies the use of rows A and C of the VMEbus P2 connector for a RACEway fabric. Using the P2 connector in this manner allows a system integrator to attach a RACEway Interlink module on the P2 backpanel, transforming the VMEbus chassis from a single

---

<sup>1</sup>Special driver features would have to be added to allow the user to assign RACEway priorities to either a device or specific transfers from a device. PCI transactions can be assigned priorities by PXB based on addresses.

bus system into a switched-fabric interconnect system. RACEway Interlink modules come in several sizes and topology options (figure 7) [6, 7].

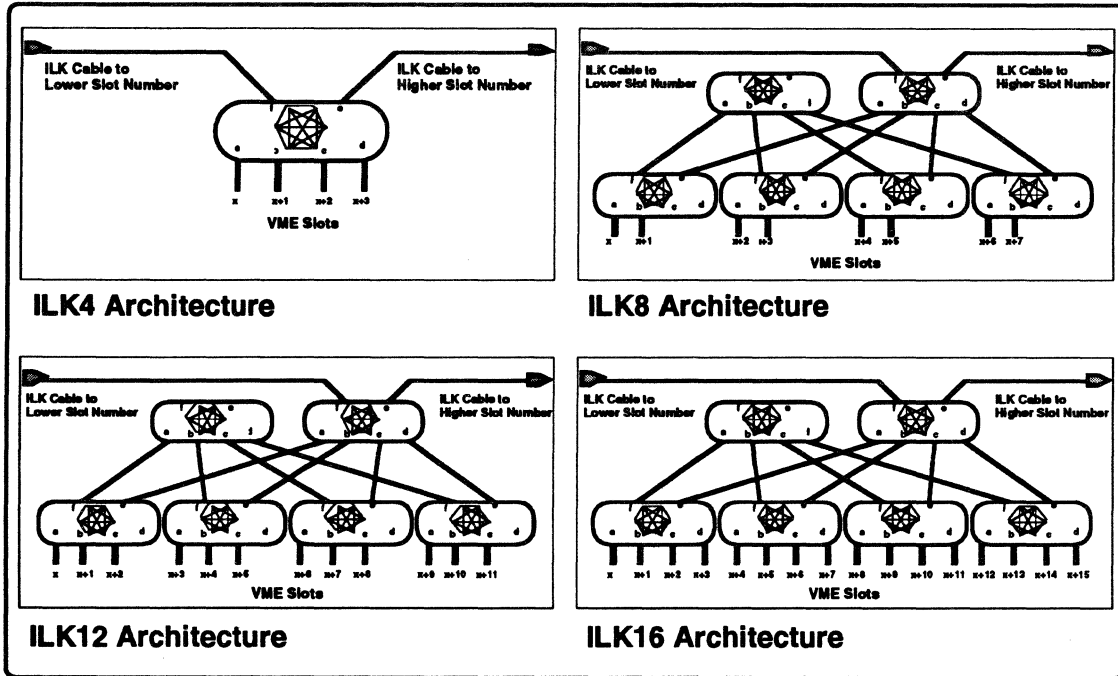


Figure 7. Example topologies of RACEway Interlink modules.

A RACEway Interlink Module consists of one or more RACEway Crossbar switches. Each crossbar is capable of 480 Mbytes/s of aggregate bandwidth. Analogous to VSB, these modules attach to the rear of a VMEbus backplane using the standard backpanel P2 connectors. All modules are expandable by adding additional modules. For example, an eight-slot solution is possible with two ILK4s, while an ILK8 offers a better bisection bandwidth over eight slots. Not shown is the ILK1 module (a simple cable with connectors) which adds one or two slots to any of the above modules or provides for a two-slot configuration.

### PMC CASE STUDY

PMC is a standard [8] that defines PCI mezzanine boards for VMEbus and other standard buses. The PMC VMEbus specification provides for a PN4 connector which brings rows A and C of P2 to the mezzanine board. PN4 permits a PXB PMC module that bridges a motherboard's PCI bus to the RACEway Interlink interconnect on the backpanel (figure 8). In this manner, RACEway Interlink can be used to extend the PCI bus of a single board computer (SBC).

The SBC in figure 8a can tie into other SBC modules. While adding multiple SBCs does not solve the problem of adding PMC locations economically, adding a full-function board just to obtain PMC locations is overkill. It is likely that a single, modern CPU is capable of servicing several PCI devices. To solve this problem, a PMC carrier board is offered (figure 8b).

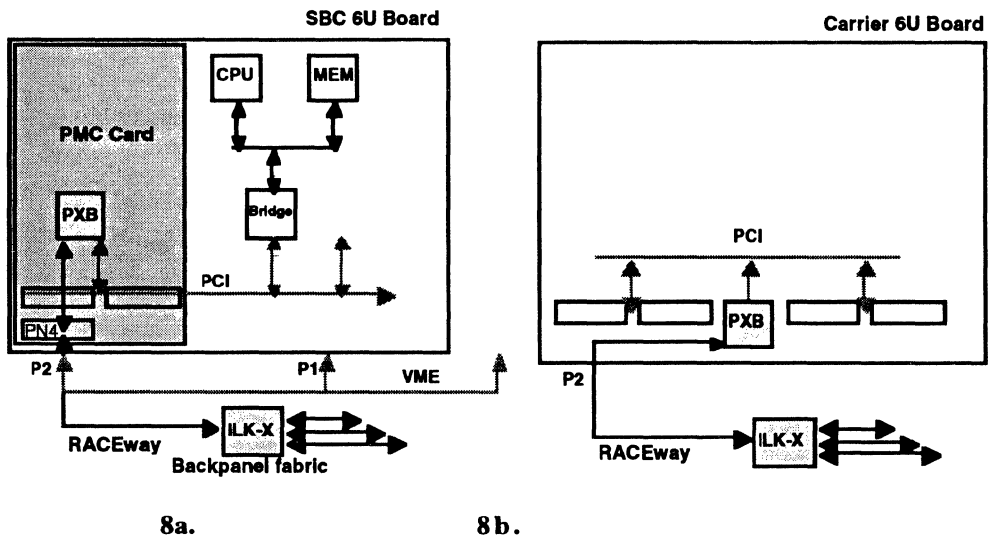


Figure 8a shows the application of the PXB on a PMC card; while 8b illustrates the use of PXB on the base-board.

Using the VME/PMC form factor with RACEway provides for up to 20-slot configurations. An illustrative example of an eight-slot configuration is shown in figure 9.

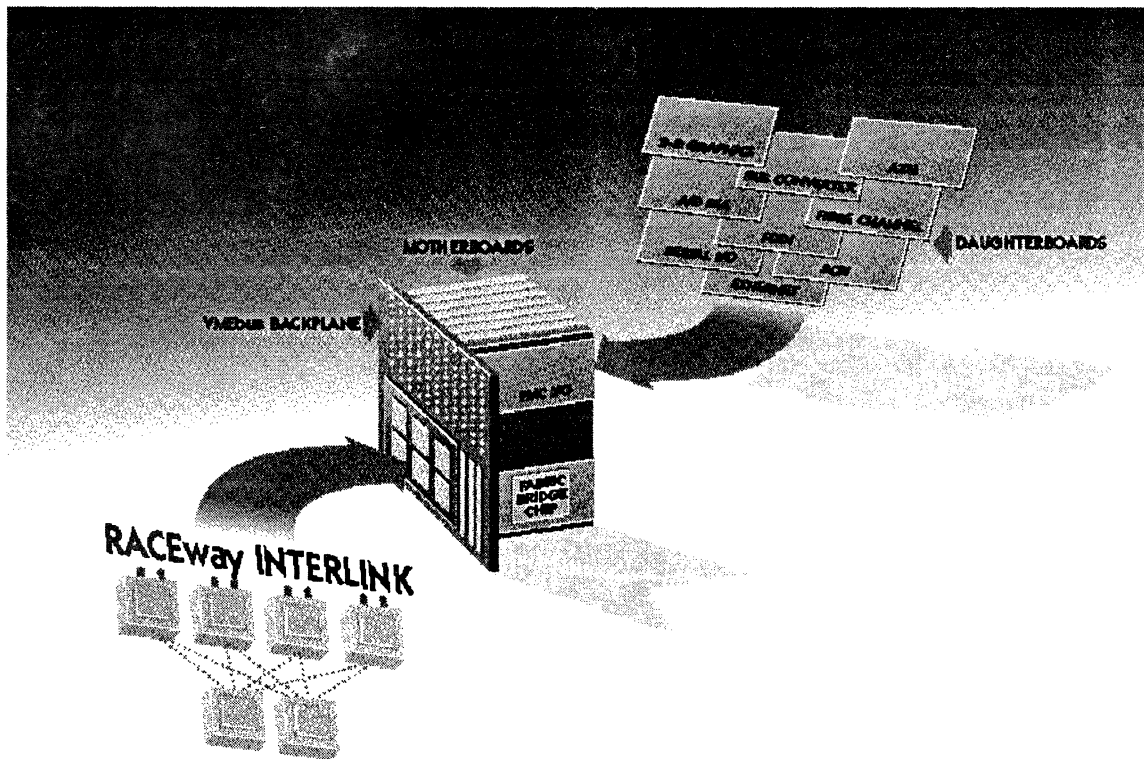


Figure 9. A PMC-RACEway case study example.

The scalable system in figure 9 uses the RACEway Interlink switching fabric to connect all the PCI buses. Off-the-shelf motherboards and PMC daughterboards provide for varied configurations to address real-time processing, digital signal processing, I/O routing, and various telecommunications and other application requirements.

## SUMMARY

While a transparent PCI switching fabric poses many technical challenges, a solution is formulated by Mercury Computer Systems and Cypress Semiconductor using the existing RACEway switching fabric. With the addition of a RACEway-to-PCI bridge chip, RACEway Crossbars will act as a transparent PCI fabric. In the future, RACEway will evolve to support 64-bit features and higher frequencies needed for supporting future PCI devices.

## BIBLIOGRAPHY

- [1] "VITA 5-1994, RACEway Interlink - Data Link and Physical Layer Specification," VFEA International Trade Association (VITA), Scottsdale, Ariz., Rev. 1.5.1, May 10, 1995.
- [2] Einstein, Thomas, "RACEway Interlink—A Real-time Multicomputing Interconnect Fabric for High-Performance VME Systems," *VMEbus Systems*, February 1996.
- [3] Blau, Bob, and Isenstein, Barry, "Introduction to RACEway Interlink," *VITA Journal*, October 1993.
- [4] Einstein, Thomas, "The RACE Multicomputer — Processors, I/O and the RACEway Interconnect," Vol. 1, Ver. 12., June 1995.
- [5] Kuszmaul, Bradley C., "The RACE Network Architecture," 1994.
- [6] RACEway Interlink Modules data sheet, DS-42-20, Mercury Computer Systems, Inc., 1994.
- [7] Cypress RACEway Interlink Modules data sheet, Cypress Semiconductor, June 1995.
- [8] IEEE 1386.1, April 1995, D2.0 Physical/Environmental Layers for PCI Mezzanine Cards, PMC.
- [9] Many of these references and additional information on RACEway and PXB are located at URL: <http://www.mc.com/technology.html>

## BIOGRAPHY

Barry Isenstein, director of strategic marketing, 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 Spring '96

## PCI Bus Switching With the PSX

Kent Dahlgren  
Director Marketing  
I-Cube Incorporated



1

Title: (LOGNAME).EPS  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/25/95) (1:15 PM)

## Overview

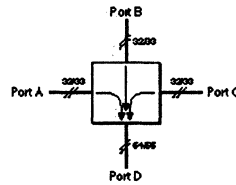
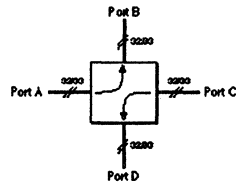
- I-Cube has developed a switched PCI evaluation platform
- Targeting shared memory applications
- Goals
  - High aggregate bandwidth
  - Scalability
  - Leveraging 32 bit / 33 MHz PCI peripherals



2

Title: (LOGNAME).EPS  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/25/95) (1:15 PM)

## Switch Dataflow Characteristics



- Peer to peer
- Transfers are evenly distributed between ports
- Typical of distributed memory parallel processing systems
- Concurrent transactions between ports
- Aggregation
- Most transfers are to or from one port
- Typical of shared memory applications parallel processing systems
- Interleaved transactions through one port



3

Title: (LOGONAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/25/95) (1:15 PM)

## Aggregation Switching for Shared Memory

- Requires one or more fat-pipes to memory
  - 66 MHz / 64 bit PCI
  - Pentium Pro
  - Other proprietary solutions
- Very sensitive to switch latency
  - Writes can be posted
  - Reads cannot



4

Title: (LOGONAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/25/95) (1:15 PM)

## Applications - Routers

Title: (ROUTER1.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/29/96) (2:35 PM)

- PCI NICs on one or more ports
  - 100 Mb Ethernet NIC = 25 MB/s traffic
- RISC CPU based routing engine
- Shared memory for packet storage



5

Title: (LOGO.NME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/26/96) (1:15 PM)

## Applications - Video On Demand Servers

Title: (VID\_SRVR.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/29/96) (2:34 PM)

- PCI SCSI controllers on one or more ports
  - Five SCSI controllers require 100 MB/s
- PCI NICs on one or more ports
  - Two STS-3c ports require 80 MB/s
- High bandwidth shared memory for stream buffers



6

Title: (LOGO.NME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/26/96) (1:15 PM)



## Applications - File Servers

Title: (File\_Server.eps)  
Creator: Adobe Illustrator(TM) 3.2  
CreationDate: (2/29/96) (1:28 PM)

- High performance I/O for network file servers.
  - 100 Mb Ethernet NICs
  - SCSI
- CPUs hang off of MP bus
- Interface with system memory via MP bus.



7

Title: (LOGNAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

7

## Shared Memory Reference Design

Title: (Sh\_Mem.eps)  
Creator: Adobe Illustrator(TM) 3.2  
CreationDate: (2/29/96) (3:55 PM)

- High speed SDRAM shared memory
- Three target-only ports
- Non-Standard switch to memory interface
- 66 MHz switch frequency



8

Title: (LOGNAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

8

## Implementation

Title: (160board.eps)  
Creator: Adobe Illustrator(TM) 3.2  
CreationDate: (2/28/96) (5:43 PM)

- 9" x 11" PCB
- Three PCI Mezzanine Connectors (PMC)
- Cypress pASIC FPGAs implement:
  - PCI port interfaces
  - Memory Controller
- External FIFO's
- 1 MB of MoSys MDRAM memory



9

Title: (LOGNAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

9

## Performance

- 33 MHz PCI bus rate for up to 6 loads
- 66 MHz FIFO / memory operating rate
- 8 PCI cycle latency (worst case) for a memory read.
- 7 PCI cycle latency (worst case) for a memory write.
  - Writes are posted
- Performance is limited by FPGAs.
  - 99 MHz memory performance is achievable in standard cell design.
- FPGA performance could be improved by incorporating FIFOs.



10

Title: (LOGNAME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

10

## PSX Switch Fabric

Title: (PSXBLK.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (12:13 PM)

- 160 I/O in a single device
- 20 Ns bus connection setup time
- 100 MHz Max data rate
- 80 MHz internal TDM Rate
- Larger and smaller versions are on the way

Title: (LOGOHOME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

11



## Scaling

Title: (Mem\_Bank.eps)  
Creator: Adobe Illustrator(TM) 3.2  
CreationDate: (2/29/96) (3:14 PM)

- Increasing aggregation bandwidth
  - Increase bus width
  - place memory banks on separate switch ports
- Increasing bus port count
  - Larger PSX
  - Slice datapath between multiple PSX devices

Title: (LOGOHOME.EPS)  
Creator: Adobe Illustrator(TM) for Windows, version 4.0  
CreationDate: (2/28/96) (1:15 PM)

12



## Summary

- Shared memory switching is a key application for switched PCI
- Crossbar switching fabrics provide a superior solution for shared memory switching.
  - Low latency
  - Scalability
  - Price/Performance (ASIC port controllers)



## Position Statement

**Session: 1A The Future of PCI**, May 1, 1996  
by Edwin Lee.

CompactPCI is a well designed combination of the PCI bus electrical and software standards and the Industrial grade Eurocard packaging standards. It is an open architecture standard, supported by PICMG, a rapidly expanding, two-year-old, manufacturer's association with over 90 member companies.

**Within 3 years, CompactPCI will become the dominant computer standard in markets that require rugged, reliable equipment.** These markets include Telecommunications, Industrial Control, Instrumentation, Medical Electronics and Military equipment.

Today the leading standard in these markets is VME bus, a well designed line of products effectively supported by VITA, one of the finest trade associations in business. But CompactPCI has a strategic advantage that no amount of effort on the part of VITA or its member companies can overcome: it can employ all the latest chips, systems and applications software, development tools, system designs, and educated customers that are developed and paid for by the more than \$150 billion/ year desktop market for PCI. These products and tools are available to CompactPCI manufacturers and Users at desktop prices. VME bus adherents must develop and pay for all their comparable products and services within the income stream of their \$1.5 billion market. That requires them to support expensive engineering and marketing infrastructures that require high margins and high prices.

VME's problem is highlighted by the fact that Motorola has recently given up its development race with Intel and will no longer upgrade its 680x0 line of CPUs, the lion's share of processors in VME products. In a sense VME bus has a tougher uphill battle than Apple, since Apple still has 7% of the desktop market to fund its battle for survival, and Apple is adopting the PCI bus in its latest desktop computers.

Ed Lee  
2/22/96



# CompactPCI™

## The Next Industrial Computer Bus Standard

presented by  
Edwin Lee  
for Pro-Log Corporation

• 1



## Presentation Overview

- What *CompactPCI* is
- What makes *CompactPCI* exciting
- *CompactPCI* and industrial alternatives

• 2



## *CompactPCI*: a combination of established standards

electrical and software standards

**PCI bus**

+

**Eurocard**

industrial packaging standards

• 3



## *PCI Bus contributes*

- Off-the-shelf Chips
- State-of-the-art software
- The best development tools
- Educated customers

• 4

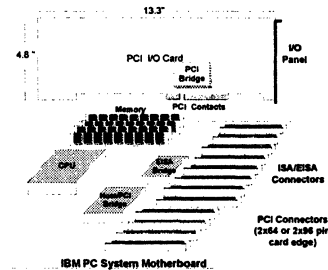


## Eurocard solves the industrial packaging challenge

• 5



## Usual layout of PCI bus



• 6



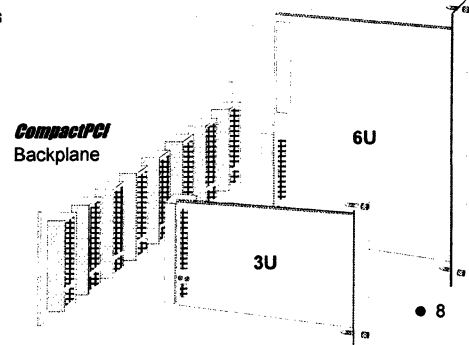
## Weaknesses of desktop packaging

- Card seating
- Time to repair/replace
  - Motherboard
  - Disassembly
- Card edge connector issues
  - pin count for grounds and shields
  - long term reliability
- Poor thermal design

● 7



## CompactPCI: Eurocard packaging



● 8



## Eurocard packaging standards

- Worldwide standards for industrial applications
  - Popularized in USA by VME bus
- Cards firmly seated, locked in place
- Plug-in System card, passive backplane
- Unblocked airflow
- Front panel insertion and removal
- Front or rear panel I/O

● 9



## CompactPCI uses Eurocard

- 3U and 6U form factor boards
  - 3U: 100 mm x 160 mm
  - 6U: 233 mm x 160 mm
- Off-the-shelf chassis, housing, and mechanics widely available from
  - Rittal, Schroff, Vector, Vero, etc.
- Shielded pin-and-socket connectors

● 10



## Connectors

- 3U card has a single bus connector
- 6U card may include an additional connector for:
  - Bridge to another *CompactPCI* bus
  - Bridge to VME, ISA, or STD bus
  - User I/O (popular in telco applications)

● 11



## CompactPCI connector

- 2 mm pin-and-socket type
  - Socket half on plug-in cards
  - Pin half on backplane
- Originally developed by Siemens for telecom applications in mid-80's
- Meets IEC-917 and IEC-076-4-101 standards
- Bellcore qualified

● 12

 **CompactPCI connector**

- Available from several vendors
  - AMP
  - Burndy
  - ERNI
  - Robinson-Nugent (new)

● 13

 **25 Rows for 32 bits**

- 110 pins: 5 pins/row (15 pins lost to keying area) + 25 pin shield

● 14

 **47 rows for 64 bits**

- 220 pins: 5 pins/row (15 pins lost to keying area) + 47 pin shield
- Includes 119 ground pins for
  - shielding
  - very low ground bounce
  - excellent noise immunity
  - low radiated emissions

● 15

 **Benefits of shielding**

- Reduced EMI
- Reduced susceptibility
- Easier to get CE certification

● 16

 **Pin Assignments for 64 bits**

- Rows 1-25 are 32 bit PCI signals
  - includes 6 reserved pins per Intel's PCI specification Rev 2.1
- Rows 26-42 are 64 bit extensions
- Rows 43-47 (20 pins) are reserved for future use

● 17

 **CompactPCI signals**

- PCI signals are identical to Intel PCI standards (including 6 spares)
- 6 additional signals
  - 1 for Push Button Rest
  - 2 for Power Supply Status (DEGRADE & FAIL)
  - 2 for Legacy ISA interrupts
  - 1 for System Slot Identification

● 18

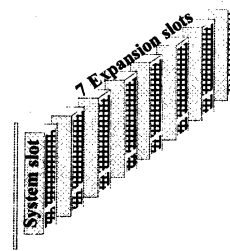


**P** Plug-in cards per system  
PRO-LOG

- 8 slots per *CompactPCI* bus module
  - 1 Host/PCI or PCI/PCI bridge slot
  - 7 expansion slots
- Up to 256 *CompactPCI* bus modules linked by PCI/PCI bridges

● 19

**P** 8 Slots per bus module  
PRO-LOG



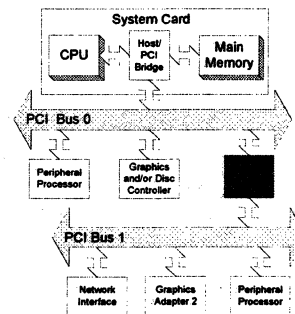
● 20

**P** 8 Slots because:  
PRO-LOG

- End-end simulation
- Choice of connector
- Impedance matching
- 10 ohm stub termination resistors on plug-in boards

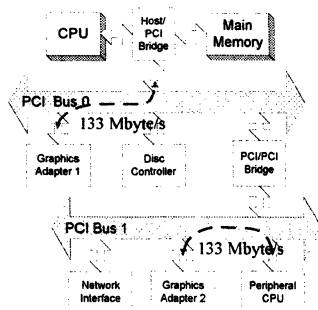
● 21

**P** 256 bus modules /system  
PRO-LOG



● 22

**P** Bridges also increase throughput  
PRO-LOG



● 23

**P**  
PRO-LOG

**CompactPCI**

**Exciting !!! because:**

● 24



## Over \$4 billion in markets

- Telecommunications
- Medical electronics
- Military off-the-shelf (COTS)
- Instrumentation
- Industrial Control

● 25



## Overwhelming cost advantages

- Chips and software
  - developed and debugged by >\$150 billion desktop PC market
  - prices driven down by low margin, high volume, cost competitive suppliers
  - readily available to industrial users, not forecast dependent
  - ...unique parts force accurate forecasts for market share and profits (a la Apple)*

● 26



## Overwhelming cost advantages

- \$1.2-1.5 billion annual revenues
  - Total sales of VME bus products (<250 thousand systems)
  - Intel's share of PCI system logic chip sets (40 million chip sets x \$30/set)

Note: Intel is only one of several suppliers.

● 27



## Overwhelming cost advantages

- Leverages customer knowledge
  - PCI bus
  - applications software
  - software tools
- Leverages third party education
  - books
  - training courses

● 28



## Very High Performance

- 133Mbytes/second @ 33MHz and 32 bits
- Upgradable to 266Mbytes/second @ 33 MHz and 64 bits
- Speeds of other buses
  - VME: ~40 Mbytes/second
  - ISA: ~ 2 Mbytes/second

● 29



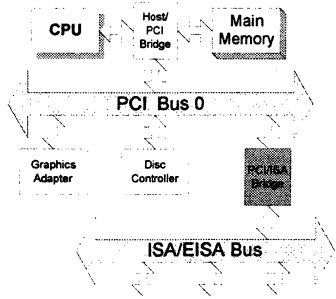
## Bridge chips link legacy buses

- PCI/ISA Bridge supports ISA I/O (Neptune, Triton, etc. chip sets)
- PCI/VME Bridge supports VME I/O (Newbridge)

● 30



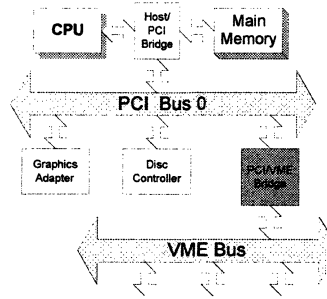
### PCI to ISA bridge



• 31



### PCI to VME bridge



• 32



### - Using legacy buses -

- It is probably in a user's best long term interests to migrate from legacy buses
  - Improved performance
  - Added features (I/O bus initiators)
  - Plug-and-play
  - Simplifies support
- ... but hybrid systems can make sense for some period of time and for some applications

• 33



### CompactPCI links to I/O buses

- SCSI
- PCMCIA
- AB Data Highway
- GE Genius
- CAN/DeviceNet
- Fieldbus
- PMC, IndustryPack
- etc.

• 34



### First to market solutions

- 6U Eurocard has more area than full sized PC plug-in card (58 sq in vs 53 sq in)
- **CompactPCI** can immediately adopt the latest PCI bus
  - chips and peripherals
  - applications software
  - development tools

• 35



### Simple to use

- Familiar concepts
- Leverages desktop PC knowledge
- Clear benefits
- Few technical details
- Plug and play

• 36



- Supported by PICMG
  - Now over 90 member companies
  - Growing rapidly
  - President: Joe Pavlat of Pro-Log
- Customers can make and/or buy
  - *CompactPCI* Standard ver 1.0 now available from PICMG

• 37



***CompactPCI* rides the  
Intel/Microsoft  
juggernaut**

• 38

**Position Statement  
Future of PCI Panel  
Session 1A**

PCI in all of its incarnations will replace the venerable ISA bus in the not too distant future. Chipset and peripheral chip manufacturers will produce their wares with only PCI, no ISA bus. Like ISA there will be many physical formats: motherboards, passive backplanes, industrial/rugged card shapes/sizes. The formats with the best defined implementation will be the ones that succeed. The leverage of common silicon will help PCI edge out old technologies, as will the software support implicit in the PCI design.

The hot topics in the future will be:

- Hot swap, especially for telecom applications, will require cooperation of the silicon manufacturers.
- Slot count--innovation is needed to overcome this limitation--needs innovation in three areas: devices, bridging or switching 'fabrics'.
- Industrial uses: Will Compact PCI replace VME?
- Bridging to other legacy buses: board limits, I/o availability, backwards compatibility.
- Multiprocessing: is there a place for anything besides what Intel is providing on chip?

**Bert Forbes**  
**Ziatech Corporation**

## PCI AND DATA ACQUISITION

Jim Fitzgerald  
Keithley MetraByte  
440 Myles Standish Blvd.  
Taunton, MA 02780  
508-880-3000  
508-823-5162 (fax)  
e-mail: fitzgerald\_jim@keithley.com

### **ABSTRACT**

After many years of struggling with the performance limitations of the ISA bus, PCI is a welcome newcomer to the world of data acquisition. In the beginning of PC-based data acquisition, the computer's resources were almost exclusively dedicated to the job of collecting and analyzing data. With the advent of GUI-based multitasking operating systems, and high performance I/O hardware such as local-area networking boards, high resolution video cards, and bus mastering hard drive controllers today's data acquisition products are in stiff competition for system resources. This paper describes the benefits of PCI to the world of data acquisition, how PCI can eliminate system resource issues and also presents some architectural and design considerations.

### **PC BASED DATA ACQUISITION BACKGROUND**

In the old days, a PC had a video card, a couple Megs of memory, a serial port, a parallel port, a hard drive and a floppy drive. There were a handful of interrupt levels and DMA channels as the mechanisms for data transfer. A user played with the base address dip switch so the software could find the board then set some jumpers to select an interrupt level that didn't conflict with something else in the computer and another switch to pick a DMA channel if the board supported DMA. The operating system was simple, i.e. DOS, and the processor was slow so it couldn't handle much anyway.

With this limited set of resources, the user had little choice but to dedicate the computer completely for the use of data acquisition and analysis. Since the system was simple, there was usually little to conflict with the process of acquiring data and a user could acquire at a rate of 100K samples per second under DMA with little problem even if the hardware had

little or no FIFO buffering. Many systems even used polling to acquire data which would completely tie up the processor but was of no consequence since very little could occur while acquiring anyway. When a user wanted to go much faster, they bought a card that had a significant amount of on-board memory, usually in the form of DRAM. Life was simple then.

As the processors got faster and the operating system more flexible, the computer was expected to do much more than simple data acquisition tasks. What with networking, multimedia, CD drives and everything else that is now standard on a modern PC, there were smaller and smaller portions left of the pie. Large FIFO's became the order of the day and still data could be lost. Bus mastering and demand mode DMA helped, but many PCs had problems with these advanced features of their non-standard ISA bus and DMA transfers were limited to 64K words before having to reprogram the DMA controller on the motherboard. Resources and bandwidth had been stretched to the limit. Micro Channel and EISA were available but neither was as widely accepted as ISA and therefore had a much smaller customer base.

### **ENTER PCI**

The promise of low latencies and high throughput on a bus with a true specification is exciting. On top of that, the PCI bus has quickly been embraced by most if not all of the industry. As opposed to Micro Channel or EISA, this is truly the bus that will replace the ISA bus. With lower latencies (in bus mastering mode) large and expensive FIFO's on the data acquisition board are not needed and with the high throughput, accumulated data is transferred before a FIFO overflow occurs. In addition, plug and play eliminates all switches on the board. Another added benefit is peer-to-peer communications across the PCI bus. This allows a data acquisition board to be

an initiator and a signal processing board as a target and visa versa. Finally, CPU independence allows selling a product over a wider range of platforms.

The following describes the architectural and design considerations for a PCI based Data Acquisition System (DAS) product.

### **DATA TRANSFER CONSIDERATIONS**

In the world of data acquisition, long latencies are intolerable. Not one byte of data can afford to be lost. Unlike with audio where small amounts of delayed or missing data may be tolerated as added distortion or in video where picture distortion or jerky motion can occur, missing data in data acquisition could mean a corrupted experiment or inappropriate action taken on the part of a control loop. This could lead to disastrous results.

There are two major data transfer considerations; the mechanism used to indicate the need for a data transfer and the mechanism for actually transferring the data. The first consideration is related to the issue of latency or *how long* it will take from the time data is available until a transfer can start. The second has to do with throughput or *how fast* a transfer can be completed.

#### **How Long**

There are several methods for determining when a data transfer is required. These are polling, interrupt and initiator bus request (used in conjunction with bus mastering).

**Polling** : This is probably the easiest method for determining when a transfer is required and is the least desirable from a performance point of view. Although polling was a viable alternative in the DOS world, this is not so in a multitasking world.

In Windows 3.1x, polling will hang the system until the desired event is reached. This is because Windows 3.1x is not a preemptive multitasking operating system. This means that your system can do nothing else but wait for data and transfer it. Not a very useful system.

In a preemptive multitasking operating system like Win95 or NT, a polling algorithm will only run while its time slice is active so the system will not

hang. However, the non-deterministic nature of this method leads to unpredictable and typically intolerable latencies.

**Interrupt** : Under today's operating systems, interrupt latencies can be from tens of microseconds ( an idle system) to several milliseconds and tend to be non-deterministic. This necessitates using very large and expensive FIFO's at best and limits total data acquisition system performance at worst.

Writing an ISR as a VXD can minimize interrupt latencies, but latencies can still suffer under a heavily loaded system especially if interrupt sharing is used.

**Initiator Bus Request** : Initiating a bus request at the lowest level of hardware handshaking results in the shortest latencies for communicating the need for data transfers. Latencies using this method range from hundreds of nanoseconds to tens of microseconds and require the least amount of data buffer memory. This method necessitates, however, that the board be capable of bus mastering and also requires on board DMA hardware. Usually this capability is built into the PCI controller chip.

#### **How Fast**

There are several methods for transferring data. These are a single read/write operation in a loop, using the "REP" instruction prefix with a read/write instruction (for an x86 processor) and bus mastering.

**Read/Write** : After recognizing that a data transfer must occur, the system CPU reads or writes data in a loop to transfer data. The problem with this method is that it requires many clock cycles for each transfer (i.e. it's slow) and uses up large amounts of CPU time.

**"Rep" Prefix** : This method is essentially the same as above except it uses a single, more efficient instruction. The problem is that too much CPU overhead is still required.

**Bus Master** : This is a background operation that does not require CPU bandwidth. Once the bus has been granted to the bus master initiator, transfers can occur at PCI clock rates (i.e.) one DWORD every 30 ns.

## Data Transfer Conclusions

After analyzing all of the options for transferring data, it is obvious that the most reliable option offering the highest throughput and lowest latencies is bus mastering. It is also the only method that frees the system processor to do what it is meant for (i.e.) processing, not transferring large amounts of data across the bus. Also, since latencies are low and transfer rates are high, smaller and slower primary buffering may be used.

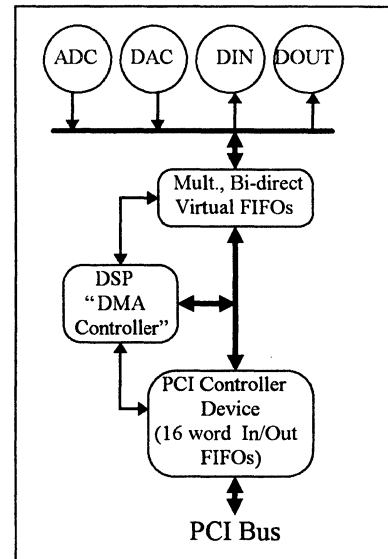
### **BUS MASTERING AND CREATING DMA CHANNELS**

Although bus mastering is possible on the ISA bus, the total throughput is still limited to about 2.5 Mega Bytes per second, conflicts with memory refresh need to be resolved and it does not work well with all motherboards. PCI, on the other hand, is designed for this type of operation.

Since DMA is not an integral part of PCI, DMA control must reside on the DAS board and utilize bus mastering as the mechanism for data transfer on the bus. One way to provide this control, is to create virtual DMA channels on the DAS board using a digital signal processing device. In this way several DMA channels are available for multiple, high speed functions on a single board. The DSP chip computes the 32 bit system address, keeps track of data in the board's primary data buffer FIFO and creates a terminal count interrupt. The DSP can also perform scatter / gather DMA when contiguous memory is not available. Since the DMA channels are created on the board, there is no longer the problem of running out of bus specific DMA resources, each DAS board carries with it its own DMA resources.

The board's primary data buffer FIFO is comprised of a dual port SRAM and several address generators in an FPGA. Multiple, bi-directional FIFO's are created in this way. The FIFO's are needed to compensate for latencies in acquiring the bus so that no data is lost. This FIFO operates at the slower board speeds as opposed to the 33 MHz PCI clock rates. This allows for a more simple and cost effective design. Data is transferred in bursts to the PCI controller device's FIFO. These smaller FIFO's of just 16 words each for input and output are able to transfer data in bursts at PCI rates.

This architecture supports an *aggregate* data transfer rate of up to 10 Mega Bytes per second. As an example, one configuration utilizes a single, 12 bit ADC converting at a up to 5 Mega Samples per second, while a second configuration utilizes a combination of four analog and digital I/O devices, *each* converting at 1 Mega sample per second. Even for these high data rates, there is no need for on board DRAM because of the low latencies and high throughput available on the PCI bus.



The above figure shows a DAS configuration of an analog and digital I/O. Each of these I/O functions are assigned an on board DMA channel. An arbiter (not shown) allows each of these functions to be multiplexed onto a port of the primary data buffer FIFO. The DSP keeps track of data in the FIFO, signals the PCI controller when a transfer is required and provides the starting address of the transfer to system memory. To initiate a transfer, first data is transferred from the second port of the primary data buffer FIFO to the smaller and faster FIFO in the PCI controller as an off line process. Then, the PCI controller device initiates a bus request and completes the transfer at bus rates while the DSP is calculating the next start of transfer address.

### **THE PCI CONTROLLER**

Another major consideration in a PCI based DAS design is the bus controller chip. There are several possible options for this device. An ASIC



(Application Specific Gate Array) may be built, an FPGA (Field Programmable Gate Array) can be designed or an off-the-shelf device can be used. The selection of a device type must be based on the following requirements: high performance, fast turnaround, low cost and ease of use. These are some of the pros and cons of each potential solution.

#### **ASIC Solution**

- + Customizable to specific needs
- + High performance
- + Lowest piece price (for large volumes)
- High NRE (on the order of \$25K to \$35K)
- Time to market
  - Very large development and verification time
  - Long lead time (proto to production ~16wks)
- Re-spins due to: PCI spec revisions, design errors and system changes
- Minimum volumes required

#### **FPGA Solution**

- + Low volumes OK
- + No NRE
- + Customizable (limited by density)
- + Can be turned into masked device (see ASIC pros and cons)
- Very high piece price (on the order of \$200 to \$300 per bus mastering device)
- Lowest performance (extra PCI clock cycles often needed)
- Large development and verification time (helped by vendor macrocell availability)
- Lowest density
- Redesign and reverification of compliance left to user with new PCI spec revisions

#### **Off-The-Shelf Solution**

- + Already designed
- + Proven PCI compliance
- + No volume minimums
- + High performance
- + Low piece price
- + Vendor redesigns the device when new PCI spec revisions occur
- Not customizable
- Will require glue logic for tailoring to specific system needs
- Need to learn device and its idiosyncrasies

#### **PCI Controller Selection Conclusion**

Given the typical production volumes for data acquisition products, the off-the-shelf device solution is the best choice. These volumes do not justify a gate array device and the prospect of continually re-spinning the ASIC due to PCI spec revisions seem daunting. The FPGA approach is not recommended because of its extremely high piece price and the increased difficulty of fitting a full bus master controller in a device without compromising performance. In addition, the off-the-shelf approach does not require as much verification time since it has already been tested for PCI compliance. Also, since vendors of an off-the-shelf controller has the highest production volumes and smallest die size (i.e. greater yields), the price of this device is low and should continue to decrease.

#### **SUMMARY**

The PCI bus is a high performance and well accepted mechanism for transferring data in the PC. This makes it a perfect choice for new data acquisition product designs. Unlike other buses that have come and gone, PCI is here to stay. This bus, however, requires a much greater design effort and attention to technical details. New architectures and design methodologies are also required. Every effort should be made to take advantage of the benefits of the PCI bus and to comply with explicit and implicit design requirements. In the next few years this bus will undergo changes. The designs we do today should try to anticipate these changes and be as flexible as possible.

#### **BIOGRAPHY**

Jim Fitzgerald is a senior staff engineer at Keithley MetraByte where he has enjoyed working for the past eight years. He has designed the digital sections of the DAS-1600 family and DAS-1800 family of flagship products and is currently in the process of developing Keithley MetraByte's next generation of data acquisition products based on the PCI bus. He has also designed several gate array devices for the company. Jim holds a BSEE degree from Drexel University in Philadelphia and has over 15 years of design experience. In his spare time, Jim brews and judges beer and plays drums in a local blues band.

## DESIGN CONSIDERATIONS FOR DATA ACQUISITION HARDWARE ON THE PCI BUS

Richard J. Burk  
Data Translation, Inc.  
100 Locke Drive  
Marlboro, MA 01752  
508-481-3700/3080 (fax)  
rburk@datx.com

### **ABSTRACT**

This paper discusses the key elements that must be addressed when designing high-performance data acquisition boards for the PCI bus. Elements of handling high-speed bi-directional data flow, such as the interface design, on-board memory and the management of the bus, are examined.

Why design a data-acquisition board for the PCI bus? Simple: Speed. If there is one sought-after commodity in many data-acquisition applications, it is data throughput. The more data that is collected, acquired, and converted to digital form, the greater the urgency to move it to where it can be stored, analyzed and displayed. PCI is a way to satisfy that need, to transfer data with the highest possible throughput in a PC environment.

Thus, PCI brings an immediate advantage to the art of data acquisition: Significantly faster data transfer to the PC's memory and display. Not only can data continuously stream to the display but PCI sets the stage for simultaneous operation of the front-end conversion as well. At the same time, the host processor is free to perform other tasks. Because speed is now at a premium, such a concurrent environment has become essential. The CPU and its memory and cache subsystems must operate independently of the peripherals bus, and information must be interchanged in the form of large blocks of sequential data.

Of course, data acquisition can benefit from PCI's many other advantages:

- The multiplexed address and data scheme is more efficient than other buses.
- There is the possibility of bus mastering, which, because of the low latency, lets peripheral cards take control of the bus in one-tenth the time of other buses.
- PCI's Plug N Play and autoconfiguration capability forever banishes jumpers and DIP switches.
- And PCI offers a known roadmap for future developments: 3.3-V operation in portable acquisition systems, bridging to outboard racks of acquisition boards--and even faster throughputs than ever.

But before PCI can be placed on board, a designer must make a number of key decisions concerning the overall architecture, the PCI interface, the analog aspects of the board, the amount of on-board control and processing, and more. Some of the decisions will be interdependent.

### **Data Rates Dictate the Design**

Functionally, the total data rate to be managed bi-directionally will determine much of the board's eventual configuration. Hinging on the anticipated data rates will be the amount of on-board memory, and whether to make the board a PCI master or slave.

That is, the PCI bus bandwidth allows board designers to create an architecture that supports the high-speed operations of not only the input and output converters, but also other board functions. For even more speed, designers can pull out all the stops by bus mastering, letting the acquisition board take control of the PCI bus.

The required digital support, then, follows from these data sampling requirements. A designer will need to pin down the specific performance levels for all the subsystems before he or she can design the PCI interface architecture.

## Master Or Slave?

Because PCI supports two options for the transfer of data, a designer of PCI data acquisition hardware has choices to make. When the required data sampling rates are slow (<1000 kS/second), the board can function adequately as a PCI slave. FIFOs can buffer the bi-directional data and the host computer, with a Pentium processor, takes care of the reads and writes of the data samples by talking to and commanding the board -- a minor task for the Pentium.

A bus master offers more flexibility, especially when support is needed for high speed data transfers. Controlling the PCI bus offers the benefits of increased throughput--theoretically, data transfers can approach 66 Msample/s. More realistically, 40 Msample/s probably can be achieved. Of course, the board complexity goes up because the PCI interface must be designed to incorporate the master logic, which can be placed in an FPGA, or a commercial off-the-shelf interface chip can be used (if one can be found with the desired performance and characteristics).

Related to the overall complexity of the hardware's architecture is the level of on-board control utilized. Implementing a high level control architecture simplifies the PCI interface and board architecture, but adds a new level of overhead that can affect the data transfer efficiency across the bus. Whereas if the designer opts for a low-level controller, the Pentium must do the "nitty gritty" work of directly accessing the on-board register data, which reduces the amount of redundant functions between the board and the PCI bus but increases the complexity of the PCI interface. As the majority of the functions required to move and control data flow and modes on the board exist within the PCI architecture, there is little need to re-create these functions on the board. This makes more work for the CPU, but eliminating the added overhead of the high-level controller significantly increases the efficiency of the data transfer. The design balance is between the complexity of the interface architecture and the aggregate bi-directional data transfer speed required for the product.

These issues impact the final design configuration and must be weighed against such factors as the availability of the resource skills needed to design a PCI interface chip in-house and the achievable performance of an in-house design vs. that of a commercial part.

## In-House PCI Interface Or Commercial?

In-house designs offer the opportunity for innovation and the ability to get to market first. Several operations within Data Translation--from multimedia to imaging to data acquisition--have taken advantage of that opportunity and, as a result, strong PCI knowledge has disseminated within the company.

Although a standard part would offer a board designer the luxury of concentrating on the acquisition or analog functions, commercial interface chips--which try to be all things to all people--do not look promising at the moment. For example, one of the design challenges is to accommodate the various chip sets found in X86-based PCs. Intel alone has developed at least three different-generation chip sets--the Mercury, Neptune and Triton --for its Pentium and P6 microprocessors. Each chip set has its own timing requirements, quirks, and design issues. Consequently, a commercial interface chip may work correctly in one machine but encounter a problem in another. The commercial interface-chip vendor may correct problems, or errata--eventually; however, with an in-house design, any problem can be resolved immediately by reprogramming the FPGA--no waiting for the reporting and correcting of such errata.

A good example of a such a potential problem is the Neptune's buffer depth--it can accept only four data transfers in a burst, that is, four at a time, with no wait states. The Triton accepts a much longer burst of data but occasionally inserts an unexpected wait for a clock or so. If the interface chip does not accommodate that wait, data can become corrupted or disappear after being put out on the bus at the wrong time.

In addition, a flexible in-house design based on FPGAs provides for quick reaction to periodic updates and changes to the PCI specification, which continually evolves over time. If the time ever comes when the interface design can be safely frozen, there is always the option of migrating to quick-turnaround, plug-compatible gate arrays for the same cost savings afforded by an off-the-shelf part.

Once the decision to go in-house has been made, the stringent electrical interface specifications narrow the field--few FPGAs can satisfy the requirements. For instance, the input capacitance had to be under 10 pF per signal.

Even so, the timing characteristics form the tougher task, especially if the designer attempts to hit the top 132 Mbyte/s PCI specification. Setup-and-hold times are 7 and 0 ns respectively, and the clock-to-output time is only 11 ns. Then there is the AC drive. Because the PCI bus is not terminated, it relies on reflective signaling. This translates to the need for an output driver with carefully controlled output impedance and drive characteristics.

Data Translation designers not only managed to satisfy the PCI bus interfacing in one FPGA, but also included much of the acquisition board's internal circuitry, including glue and triggering logic, the interface to the counters and DACs, the decoding and digital controls for the analog circuits, and the PCI configuration registers.

Depending on the relative speeds involved, it is the designer's decision to bus master or not and how to move data. A complicating factor is that mastering can be designed in on the input or output side. The input side may be going so fast, it needs to be a master; however, other than a slight inefficiency, there may not be a problem in being a slave on the d/a side, assuming there is enough throughput on the bus. That is so because the d/a converter may be working at only 200 kHz, certainly not megahertz. Other situations may dictate mastering on both sides.

From the hardware design standpoint, the level of bus mastering implemented is a complex issue; mastering in both directions takes more logic to ensure that multiple masters cooperate among one another. All things being equal, a designer may opt for a dumb slave with no memory because it is the "cheap and easy" way to go. Like everything else, it is a tradeoff.

### **How Much Memory?**

In designing a PCI board, one objective is to reach the best balance between bus access and on-board memory capacity. For example, taking over the bus means less memory buffering is needed, and there is less overhead. The board can wait until it accumulates a number of samples, then burst data and give up the bus. That calls for a small FIFO buffer within the interface FPGA. It saves money because the improved bus management obviates the need to put expensive memory on the board. The PCI specification limits the time a board can be held off the bus; otherwise, a large buffer would be necessary to back up data when other activity takes place, such as on the ISA bus. The moral: By controlling the bus, not only is throughput improved and less memory is needed on the board, but the host processor is off-loaded to service other tasks.

Take, for example, an acquisition board that is set to sample analog input at a 3 to 5 MHz rate. As a bus master in that application, almost no memory is needed. Certainly, whatever memory is still required is much less than if operating on the ISA bus. Data flows at higher speed, in a more-efficient, more-continuous manner. There are no significant latencies. And the pace is more predictable because the board is not held off for interminable periods by other boards within the system.

### **Other Considerations**

One thing to watch for as boards speed up: Meeting the European CE (emissions and susceptibility) and EMI specifications. The faster speeds of PCI bus operation call for a more careful layout of board traces and components to prevent interference, crosstalk, and the like. Filtering and shielding may also be necessary.

### **Challenges For the Future**

Only time and money prevent a designer from expanding horizons. For instance, once it has been decided to put a controller on board, why not let the controller also perform some digital signal processing? With, say, a C52 serving as the controller, the board can be made even more independent from the CPU. And the door then opens to PCI-based real-time control and test-and-measurement applications.

The bridging aspects of the PCI bus are intriguing. Thanks to PCI bridge chips, multiple acquisition and other boards can be designed into a rack or other system or, as computers appear with more PCI slots, into a PC itself. Many more channels can be gained and the aggregate speed goes up. A multiboard design may move twice as much data so the boards efficient utilization of the bus interface will be important. Of course, that may affect the memory and other requirements.

### **Conclusions**

The PCI interface bus has eliminated many of the problems associated with the ISA bus and has widened the technology horizons for data-acquisition boards. Throughput has especially benefited. A thorough understanding of the PCI specifications is necessary to achieve the best performance and feature set within a specific board.

### **Author Biography**

Richard Burk joined Data Translation in mid-1995 as senior hardware design engineer, responsible for leading the company's data acquisition board design projects.

Burk has extensive experience in system concept, design simulation and integration. While at Concurrent Computer Corporation as principal engineer, he evaluated superscalar microprocessors for next generation computers. Previously, Burk worked at Raytheon as senior design engineer, where he designed several VMEbus controller modules and IO controller ASICs for use in systems such as electrical to fiber optic translation, data communication and target simulators.

Burk graduated Magna Cum Laude with a Bachelor of Science degree in Electrical Engineering from Tufts University in Medford, MA.

## THE IMPACT OF PCI ON THE TEST AND MEASUREMENT INDUSTRY

Arthur Ryan

National Instruments

6504 Bridge Point Parkway, Austin, TX 78730

(512) 433 - 8845 / 8641 (fax)

e-mail: ryana@natinst.com

### *Abstract*

The use of personal computers for test and measurement applications is gaining more and more acceptance in the test and measurement market. However, a major hindrance to the use of personal computers in certain test and measurement applications has been the lack of a standard, high-speed bus architecture. PCI is an enabling technology that dramatically increases the domain of test and measurement applications suitable for PC-based test and measurement. Applying PCI to the test and measurement industry presents several technical and architectural issues that must be resolved to fully realize the potential of PCI. This paper discusses the impact of PCI on the test and measurement industry and presents National Instruments technology that addresses the technical and architectural challenges with integrating PCI to the test and measurement industry's formal and de facto standards, such as data acquisition, VXIbus, IEEE 488, and virtual instrumentation.

### *Introduction*

The Peripheral Component Interconnect (PCI) standard is profoundly affecting the computer industry; the influence of PCI is spreading to other computer-related areas, such as the test and measurement industry. Test and measurement applications and products typically use three primary technology areas. The first is the IEEE 488 standard, commonly referred to as the General Purpose Interface Bus (GPIB). GPIB is the protocol of choice for connecting remote instrumentation to a PC. The second is the VME eXtensions for instrumentation (VXI) bus while the third is computer-based data acquisition. PCI impacts all three areas. The high bandwidth of PCI provides opportunities for PCs to directly handle more applications based on GPIB, VXI and data acquisition. To fully use all PCI advantages, it is important for test and measurement vendors to provide PCI products that address the needs of VXI, GPIB, and data acquisition. Having this global view helps customers take full advantage of PCI capabilities in their applications. This paper examines the technical challenges and possible solutions for applying PCI to VXI, GPIB, and data acquisition (DAQ) products and applications.

### *PCI Encompassing More Test and Measurement Applications*

It is common in the test and measurement industry to link a computer to a set of instruments or sensors to stimulate and check a device under test. The computer, by its very nature, can perform many different roles in these systems. The flexibility of the computer has given rise to virtual instrumentation, in which a computer running appropriate software can be programmed to mimic a more expensive instrument, customize the interface of an instrument, or perform a function for which there is no commercially available product.

Applying virtual instrumentation to a problem includes evaluating the data bandwidth and processing requirements. Some applications involve small amounts of data or low-bandwidth data streams while other applications are more data-intensive. With the high bandwidth of PCI, computers can handle more data-intensive tasks than was previously possible with other bus standards, such as ISA or NuBus. The acceptance of PCI has also brought with it increased processing power in the form of the Pentium, Pentium Pro, Alpha, and PowerPC processors.

While PCI addresses the data bandwidth issues in virtual instrumentation, it also addresses a more subtle problem in computer architecture. Although many test and measurement customers use a “Wintel” computer, a sizable group of users do not. With more computer vendors adopting PCI, computers will share common architectures and differ only in the processor and operating systems they support, making it easier to change computing platforms if system requirements change. With PCI, a variety of computer system vendors can serve test and measurement customers who were previously locked into one vendor’s computer system.

Getting the most out of PCI for test and measurement applications requires a great deal of architectural planning. Customer requirements differ when using VXI, GPIB or data acquisition. Therefore, it is important for vendors to address all these issues up front to provide customers with PCI test and measurement products that realize all the potential of PCI and satisfy customer needs.

### ***Applying PCI to VXI Products***

VXIbus is a superset of the VME standard that includes interrupt and triggering protocols suitable for instruments. VXI instruments exist as separate cards inserted into a backplane. All VXIbus systems require some device to act as a controller that configures and operates the other VXI devices. Three types of controllers exist. The first is an embedded computer that is plugged into the backplane just like any other VXI instrument. A second method connects a personal computer to the VXI backplane using one GPIB hardware interface installed in the computer and another on the VXI backplane. A third method connects a personal computer to the VXI backplane using the MXIbus, which provides a memory mapped architecture with higher bandwidth than GPIB. PCI can improve VXI controller performance, but to do so, a PCI to VXI interface must address several technical issues.

VXIbus defines more address spaces than PCI.

VXIbus uses big endian byte ordering while PCI is little endian.

VXIbus and PCI have significantly different signal protocols.

VXIbus and PCI support multiple masters but have different latency and arbitration requirements.

VXIbus has multiple address spaces that are 64 KB, 16 MB and 4 GB long. To map PCI bus cycles to the VXIbus, interface cards require configurable window hardware to map specific blocks of PCI address space to specific VXI addresses and address spaces. Further, because VXIbus handles multiple masters, a PCI/VXI bridge needs window hardware to map portions of VXI address spaces back to PCI.

The PCI bus employs little endian byte ordering; the VXIbus is big endian. Simply swapping data bytes does not adequately address the byte ordering problem. For example, if a 32-bit quantity is comprised of four 8-bit characters, then the bytes should be swapped. However, if the 32-bit value is an integer, then the value should not be swapped. Incorporating byte transposing hardware into a PCI/VXI bridge alleviates control software from much of the byte ordering task, but is important to implement the byte transposing hardware so it can be disabled by control software when necessary.

VXIbus employs different signal conventions than PCI. PCI master and slave devices indicate when they are ready to transfer data; it is implied that data transfers whenever both devices are “ready” on the rising edge of the bus clock. With this approach, master and slave devices can decide independently when they are ready. VXIbus, however, uses a more strict initiate/respond protocol in which the master device asserts a *data strobe* signal to begin a transfer and the slave responds to the *data strobe* by asserting a *data acknowledge* (DTACK). A slave cannot assert a DTACK, however, until a master has asserted a data strobe; so, the slave device behavior is dependent on the master device.

The main implication of the differences in signal protocol is that a PCI cycle will eventually have to be converted to the more restrictive initiate/respond protocol on the VXIbus as the cycle crosses a PCI/VXI bridge device. Write post buffers help alleviate the problem for write cycles, but on read transfers, a PCI to VXIbus bridge needs to support PCI delayed transactions.

Latency and arbitration differ between PCI and the VXIbus. VXI device response times can exceed PCI latency requirements. The initiate/respond protocol issue previously discussed can reduce performance as the internal buffers are filled over long transfers. A good way to address these problems is to support DMA controllers within the PCI/VXI bridge. These DMA controllers, equipped with FIFO buffering, can be optimized to perform VXI and PCI cycles efficiently. Further, the host CPU does not have to deal with moving data over the PCI/VXI bridge, but can delegate that task to the DMA controller. The end result is the PCI/VXI bridge device can act as a "good citizen" on both the VXIbus and PCI.

### ***Interfacing PCI to GPIB***

GPIB is a long-established standard in the test and measurement industry. GPIB interface products are available from many vendors to connect virtually every type of computer to GPIB equipment. Because PCI is such a significant standard in the computer industry, it is important to provide a PCI interface to GPIB. Indeed, some new computers, such as Apple's Power Macintosh, provide only PCI slots; in these cases, it is important to provide PCI/GPIB support so that customers can take advantage of these high-performance, PCI-only computers.

The GPIB bus is slow by comparison to PCI and is only eight bits wide. Simply porting an ISA-style GPIB card to PCI would slow down performance over the PCI bus. A PCI to GPIB interface should use the full bandwidth of GPIB and prevent its lower performance from degrading PCI performance. Several means exist to address this problem. First, a PCI to GPIB interface should provide write post buffering. This buffering helps decouple PCI and GPIB. Second, supporting delayed transaction, as described in Version 2.1 of the PCI specification, prevents GPIB read cycles to slow instruments from occupying PCI.

A third method to improve system performance is to support byte packing in the PCI to GPIB interface. Such capability can convert a single 32-bit PCI cycle into individual 8-bit GPIB cycles. Hence, PCI does not have to routinely carry 8 and 16-bit cycles. Finally, a DMA controller on the PCI/GPIB interface would use PCI efficiently for large blocks of data.

PCI offers further benefits to GPIB performance. Recently, a high-speed protocol was added to the GPIB standard called HS488. The maximum throughput of HS488 is 8 Mbytes/s. The throughput of HS488 is higher than the typical performance for ISA and other common bus standards. PCI, by contrast, can use all of the potential bandwidth for HS488.

### ***PCI for Data Acquisition***

PCI has clear benefits for data acquisition. The average bandwidth on older bus architectures, such as ISA, is 1-2 Mbytes/s. PCI, however, offers average system performance of 20-30 Mbytes/s currently; this number is increasing with the introduction of more efficient PCI devices and chip sets. Assuming 16-bit sampling, PCI's bandwidth could support 10-15 MHz sampling applications without the need for memory on the card.

A key point to remember about applying PCI to data acquisition problems is that PCI is not necessarily appropriate to all data acquisition applications. It is important to assess the data bandwidth requirements of the customer before throwing a PCI card at the application. If the customer requires a single DAQ



board sampling at 40 kHz, then an ISA-style PCI card may be more than sufficient for the application. In such applications, the update rate of the DAQ card is slower than either the cycle time on PCI or ISA. In the case of low bandwidth sampling, a customer is better off using one of the many ISA slots on typical "Wintel" machines instead of using one of the few PCI slots.

Developing a PCI DAQ card, even for the 100-300 kHz sampling range, requires more than simply porting an existing ISA bus design over to PCI. Porting such an ISA design implies that the DAQ card will operate as a PCI slave device, but developing the slave interface will require many of the architectural features associated with the GPIB products. This similarity in architecture is not surprising because the bandwidth of a 100-300 kHz DAQ card is similar to the GPIB. PCI slave DAQ cards can benefit from features such as byte packing, write post buffers, and delayed transaction support.

Many PCI DAQ products on the market handle PCI slave-only functionality, but to fully realize PCI benefits, a PCI DAQ card should have PCI master capability. Such master capability would probably be driven by a DMA controller on the DAQ card. Bus master capability uses the PCI bus more efficiently than relying on the host processor to move the data. Bus master capability is almost a requirement for sampling rates above 10 MHz because the processor would have trouble moving data from the card while performing other operations. Bus master capability also becomes important in systems with multiple data acquisition cards whose aggregate bandwidth requirements approach 10 Mbytes/second.

### ***Implementing PCI Capability for Test and Measurement***

National Instruments has been a major test and measurement vendor for many years. Thus, it was important for us to provide PCI products to our customers. The goal of the PCI product development was to develop a series of PCI cards that addresses the key technology foundations of the test and measurement industry (specifically, VXIbus, GPIB, and data acquisition). Because National Instruments focuses on T&M applications, general off-the-shelf ASICs didn't provide the special features we needed.

National Instruments developed an ASIC that could address specific customer needs for test and measurement PCI products. The resulting ASIC is a multiport device that includes ports to MXI, PCI, and VXI. It also incorporates a general-purpose IO port for data acquisition and GPIB applications. The ASIC includes DMA controllers that can transfer data at a peak rate of 132 Mbytes/s. The DMA controllers themselves operate in several different modes. These modes can perform operations such as ring-buffer transfers, large contiguous block transfers, and noncontiguous transfers by reading and executing DMA instructions stored in system memory. The noncontiguous transfer capability is especially important in virtual memory operating systems, such as Windows and Mac OS.

All of the National Instruments PCI cards use the ASIC, thereby providing a unified PCI architecture for these cards. Providing a unified PCI architecture for test and measurement products benefits both developer and user. For the developer, it means that a single interface can be used on a variety of computer platforms, reducing time to market. Because the ASIC directly addresses the needs of test and measurement, the products can also attain maximum bandwidth for GPIB, VXI, and data acquisition. Users automatically have a migration path to new computer systems. This saves money, because the user doesn't need to buy a new board if he or she wants to use another vendor's computers. The user automatically has more computing platforms to choose from and the users' tests run faster because of the improved bandwidth.

# Chip Makers Roundtable

## Programmable Logic Implementations of PCI

*David Ridgeway, Xilinx*

This tutorial will cover implementation of custom PCI compliant I/F solutions. The focus will be on the design considerations for PCI logic design, PCI configuration, back-end interface issues and maximizing data throughput. The presentation will also review the use of Xilinx PCI Interface Modules to increase design efficiency and accelerate the engineering/manufacturing of new products.

## POSITION STATEMENT for Chipmakers Rountable

MIKE SALAMEH  
President  
PLX Technology

### PCI is not just for PCs: Embedded Systems Migrate to PCI Architecture

PCI is a well-accepted standard in Personal Computer systems. The architecture is also rapidly gaining popularity in embedded systems, replacing proprietary buses. PCI offers many advantages to hubs, routers, printer engines, set-top boxes and other embedded systems that require a high-speed, low latency backplane.

Bandwidth is the most obvious, but not most important, benefit. With data rates of 132 (32 bits) and 264 (64 bits) Megabytes per second, PCI is the first standard personal computer bus to deliver bandwidths comparable to proprietary buses. ISA, EISA and Micro Channel, which ranged in throughput from 5 to 40 Megabytes/second, were rarely attractive outside of the PC architecture world.

Perhaps more important than bandwidth are the other benefits that embedded systems designers gain from using PCI: wide availability of low-cost I/O silicon (e.g. LAN, SCSI and video controllers), a proven high integrity bus standard, and compatibility with other manufacturer's hardware.

Most suppliers of high volume I/O and graphics chips now provide a PCI interface, and sometimes only a PCI interface, for their newest products. This makes it simple and inexpensive to connect the component to the embedded system if the embedded system uses PCI. Conversely, if the embedded system does not use PCI, it incurs a cost, and sometimes a performance penalty when connecting to a PCI I/O chip.

Many embedded system designers believe, and correctly so, that they can design a higher performance and lower latency proprietary bus system that is better tailored to their products than PCI. However, the faster time to market of using a proven architecture like PCI often outweighs the marginal performance benefit of the proprietary bus.

"Open Embedded Systems" are now much easier to realize with PCI. Already manufacturers of communications equipment are taking advantage of PCI by using third party PCI adapters (for example T1 or ISDN boards) in their embedded systems. Using these adapters saves them the time and expense of designing all the embedded system peripheral devices in-house, allowing them to focus their resources on their area of expertise.

#### **PLX and PCI Embedded Systems**

PLX Technology is the leading supplier of interface chips for PCI embedded systems and adapters. PLX PCI chips are used in hundreds of such products connected to many varieties of embedded CPUs including the Intel i960® processors, 186, 386, 486, 68K, PowerPC, R3000, ARM, Inmos transputer and many DSP and intelligent I/O controllers.

# PCI Performance Analysis for High-Speed Networking

**Integrated Device Technology, Inc.**

**Peter N. Glaskowsky, Senior Engineer**  
2972 Stender Way, Santa Clara, CA 95054  
408-988-5636 png@ldt.com



Copyright 1996 IDT, Inc.



## Goal of this presentation

- **To show how to analyze the in-system performance of IDT's NICStAR ATM controller, based on:**
  - **NICStAR system requirements**
  - **NICStAR operations**
  - **System behavior**
  - **Other software and hardware issues**
- **This same process may be used to analyze the performance of any PCI device**



2



## NICStAR requirements

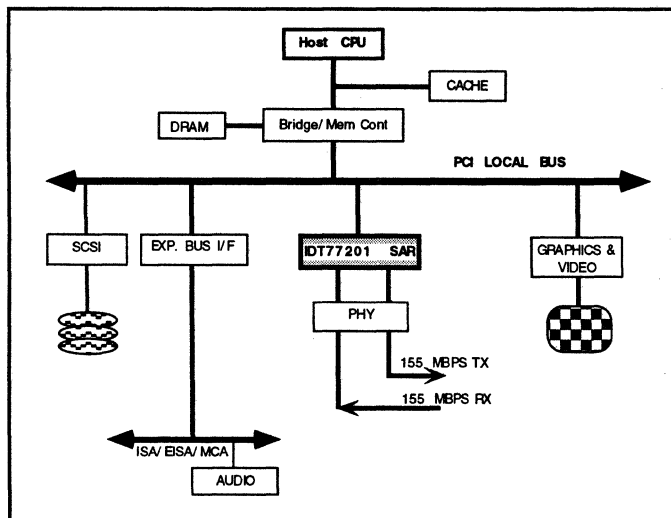
- **NICStAR interfaces between PCI and a full-duplex OC-3 (155 Mbps) ATM network**
  - **Handles ATM processing**
    - **Segmentation and Reassembly for ATM Adaptation Layers 0, 1, 3/4, and 5**
    - **Also supports 25 and 52 Mbps ATM**
- **NICStAR is a 5V, 32-bit, 33 MHz PCI device**
- **Acts as a PCI target for configuration and control**
  - **control registers can be in I/O or memory space**
- **NICStAR also operates as a PCI master to transfer its own transmit and receive data over PCI**



3



## Example NICStAR PCI System



4



## Bandwidth analysis

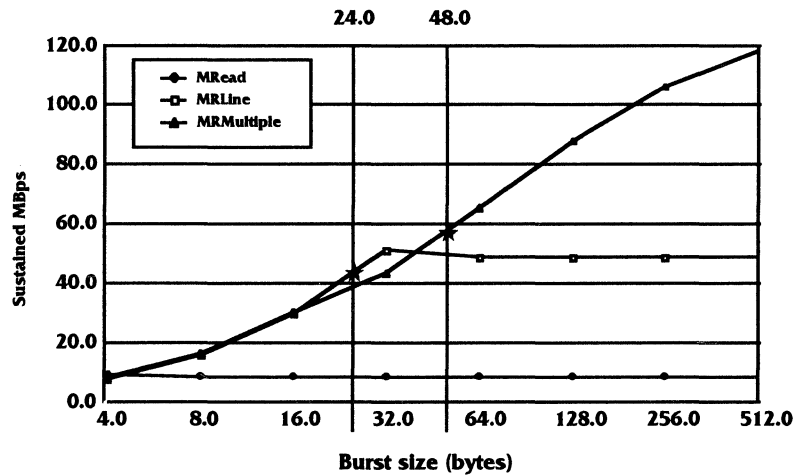
- We'll begin with cell data transfer over the PCI bus
- 155 Mbps ATM actually yields about 135 Mbps after SONET and ATM layer overhead
  - That's about 17 MBps each way, or 34 MBps of data to be transferred over the PCI bus
- This data is transferred in 12 word bursts (max)
  - Some chipsets break transfers on cache line boundaries (average 6 words per transaction)
- Let's plot this against representative PCI bandwidth curves and see where we are...

**PCI**  
LOCAL BUS

5



## Burst Read Performance

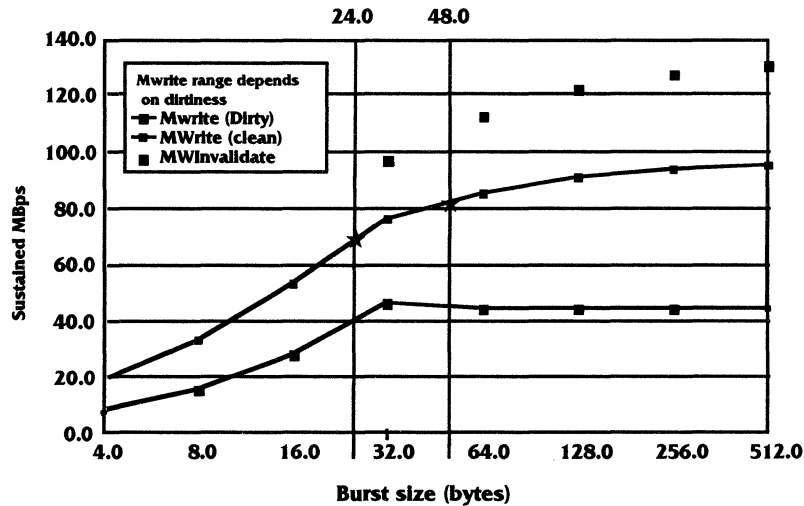


**PCI**  
LOCAL BUS

6



## Burst Write Performance



7



## Cell data transfer conclusions

- **Don't put too much trust in these curves!**
  - They are representative of current chipsets, but older chipsets can be much less efficient
- Note that longer read bursts can use the more efficient Memory Read Multiple transaction
- We can now calculate bus utilization:
  - We need  $(17/42) + (17/70) = 64\%$  of the PCI bus for our 34 MBps of data
- Therefore, cell data transfer alone does not exceed available PCI bandwidth



8



## Latency analysis

- We also have to meet NICStAR's latency specs
- On average, we transmit or receive a 53-byte ATM cell every 2.8  $\mu$ sec at 155 Mbps
- The NICStAR transmit FIFO holds 12 cells of transmit data, so the maximum tolerable PCI read latency is about 33  $\mu$ sec (non-cumulative)
- The NICStAR receive FIFO is in off-chip SRAM, and holds 315 cells, so the maximum tolerable PCI write latency is about 890  $\mu$ sec (non-cumulative)
- Most PCI systems meet these requirements easily



9



## Driver overhead

- The device driver will communicate with NICStAR to open or close the ATM virtual connections
- The driver must also work with the operating system and applications software to allocate and release buffer space in host memory
- Interrupt sharing imposes additional overhead
  - Drivers get called when they have nothing to do



10





## Interrupts and queue maintenance

- **NICStAR interrupt overhead varies depending on type of traffic and NICStAR configuration**
  - **Could be multiple interrupts per cell if NICStAR must assemble a cell from all over host memory**
  - **ATM-aware applications can transfer very large blocks of data with only one interrupt per block**
- **NICStAR maintains status queues in host memory**
- **The host CPU maintains buffers for transmit and receive data, and notifies NICStAR where they are**
- **Only PCI writes are needed for this maintenance**
  - **Posted writes reduce PCI bus overhead**



11



## Protocol overhead

- **Current network protocols have a lot of overhead for the protocol stacks and drivers**
- **Users think of “wire rate” as a guideline at best**
  - **2-4 Mbps over 10 Mbps Ethernet is common**
- **IP LAN emulation over 155 Mbps ATM will inherit all the same overhead, plus more**
- **(Today, the bottleneck is here, not in the hardware)**



12



## Limits to performance

- **The more data being transmitted or received by NICStAR, the more work there is to be done by the host CPU and other system resources**
- **At some point, you may run out of bandwidth on the CPU, or main memory, or the PCI bus**
- **The limits on each system resource must be considered when integrating any device like NICStAR into the system**



13



## Applications– server vs. workstation

- **Servers will have a rough balance between transmit and receive traffic, requiring full duplex operation**
- **Workstations will typically spend more time receiving data, more like half-duplex operation**
- **Receiving also uses less PCI bandwidth and tolerates more latency than transmitting, so is easier to support**



14



## Application issues

- Applications which require the data to pass over the same PCI bus twice can be a problem
  - For example, network » host memory » hard disk
  - 32-bit 33 MHz PCI simply isn't fast enough to support this with full-duplex 155 Mbps ATM
  - For workstation applications this is no problem
- Applications which decompress the received data (such as MPEG decoders) are even worse



15



## Towards better software

- Support native ATM protocols in drivers and applications instead of emulating old LAN protocols
  - New data types allow more efficient CPU use and more efficient use of the wire
  - Therefore more net bandwidth for the user, but...
  - ...Therefore more traffic on the PCI bus
  - ...Therefore more user demand for even faster PCI implementations



16



## Hardware to avoid

- **Older chipsets**
  - Some don't support PCI bursts
  - Many are bad at sharing host memory with PCI
- **Important peripherals on ISA (or EISA, or MCA)**
  - Hard disk controllers, video, etc.
- **Non-bus-master IDE controllers**
  - Some of these will tie up the CPU for hundreds of microseconds
- **Slow PCI devices on the same bus**



17



## Towards better hardware

- **Today:**
  - NICStAR has specific support for video data, MPEG bitstreams, etc.
  - Use new chipsets, PCI bus-master disk controllers, multiple PCI host bridges
  - Avoid sharing interrupts
- **Tomorrow:**
  - Wider/faster PCI, more concurrency, longer bursts, more use of Memory Write and Invalidate
  - Avoid use of host memory to buffer disk data
  - Faster ATM protocols: OC-12 (622 Mbps), etc.



18



## Summary

- **32-bit, 33 MHz PCI is fast enough to support current high-speed networking devices like IDT's NICStAR ATM controller**
- **NICStAR is ideal for workstations, especially if the software can take advantage of ATM protocols**
- **NICStAR can be used in servers but the system requirements are much more demanding**
- **OEMs and system integrators must consider many factors to ensure good system performance**



19



## For more information on IDT's PCI-bus products:

- **NICStAR and other ATM products**
  - **Michael Olsen, IDT ATM marketing:**
    - **408-944-2153**
- **The R4761/R4762 chipset with PCI host bridge**
  - **For IDT's R4600/R4650/R4700 MIPS RISC CPUs**
  - **Jamal Haider, IDT RISC marketing:**
    - **408-492-8623**



20



## THE PCI MULTI-FUNCTION DEVICE: BENEFITS AND DESIGN CONSIDERATIONS

Margit E. Stearns  
Symbios Logic, Inc.  
1635 Aeroplaza Drive  
Colorado Springs, CO 80916  
719-573-3228/3037 (fax)  
e-mail: margit.stearns@symbios.com

### **ABSTRACT**

As system performance becomes increasingly important in PCs today, we are seeing a movement away from the legacy buses towards the PCI local bus as the bus of choice. As that switch takes place, OEMs are faced with one of the most serious limitations of the PCI Bus: the small number of loads on the motherboard and the number of expansion connectors that can be supported. Other factors restraining the system designer include limited real estate available on the motherboard, and in some cases, a constraint on the number of bus masters supported by the chip set. One solution to these design issues is the PCI multi-function device, which may contain two or more functions within one device, while presenting only one load to the PCI bus.

This paper will define the multi-function device, and highlight the design rules and issues associated with its use.

### **WHY CONSIDER A MULTI-FUNCTION DEVICE?**

It is generally recommended that each PCI bus in a system have a maximum of ten loads. A load is defined as a PCI device on the motherboard. When the device is on an add-in card, the connector on the card is considered one load, and the PCI device on the card is considered a separate load. Therefore, a PCI add-in card counts as two loads. If more than one load is desired behind the connector, a PCI/PCI bridge must be used on the card. Examples of common PCI loads in a desktop, workstation, or server include Ethernet, SCSI, Host/PCI bridge, PCI/Legacy bridge, video, sound, and E-IDE.

A typical PCI system design has two or three card slots, and two or three PCI loads on the motherboard. In some designs, however, the limitation on the number of devices and add-in cards can present a problem. These designs may be high-end servers where many features are demanded by the customer,

or a workstation or desktop where the customer is very sensitive to the number of card slots available for system expansion.

Many systems designers are constrained by the physical dimensions of the chassis and motherboard. In a feature rich design, it is difficult to fit all of the required components on the motherboard, and still offer the number of slots that customers demand.

Another restriction may be the number of bus masters allowed by the chip set in the system. Some processor chip sets limit the number of bus masters to four, for example, because that is the number of Request/Grant (REQ#/GNT#) pairs that are available from the arbiter.

All of these design constraints lead to the PCI multi-function device as an effective solution. On the motherboard, the multi-function device represents one load to the bus, and can contain from two to eight functions. An add-in card with a multi-function device represents two loads, but contains from two to eight functions instead of one for a single-function device. By integrating more than one function on a single chip, it is not necessary to use a costly PCI/PCI bridge on the add-in card. This results in the freeing up of valuable PCI slots in the system, optimizing the number of features offered to the end user.

The multi-function device takes up less space on the motherboard than multiple single-function devices, leaving room for more features. And the multi-function device has only one REQ#/GNT# pair connection, and represents one bus master to the central arbiter, allowing the system designer to add more bus masters to the system.

These powerful attributes of the multi-function device give the system designer more flexibility and the opportunity to offer more features in the system, which leads to a competitive advantage for the OEM. The multi-function device also offers improved system reliability over a multi-chip solution, and

allows the OEM to buy multiple functions from the same vendor, simplifying the buying and design processes.

The remainder of the paper will be devoted to specifics on how to design a multi-function device into a system. The main considerations when using a multi-function device in a design are configuration accesses, interrupts, system BIOS, and arbitration.

### ***ACCESSING A MULTI-FUNCTION DEVICE ON THE PCI BUS***

A PCI device can contain one to eight configuration spaces. A single-function PCI device has one configuration space, and a multi-function device has between two and eight. A bit in the device's "Header Type" configuration register, offset 0Eh, defines whether or not the chip contains one or more functions. Bit 7 of the "Header Type" register, if set to 0, defines a single-function device, and if set to 1, defines a multi-function device.

A multi-function device, like a single-function device, uses one of the 21 individual Initialization Device Select (IDSEL) lines to determine whether or not to respond to an access. Assertion of a device's IDSEL during the address phase of a configuration access selects the device for a configuration access.

During a configuration access, AD[10::8] identify the function number of the configuration space within the device. For a single-function device, AD[10::8] is always [000]. Multi-function devices must always implement Function 0 (AD[10::8] = [000]), but may assign other functions in any order. For example, a two-function device must implement Function 0, but can choose any of the other function numbers (1-7) for the second function. Any function number that is not implemented should be ignored by the device; i.e., the device should not assert Device Select (DEVSEL#).

Whereas a single-function device only looks at the IDSEL line to determine whether or not it is accessed, a multi-function device uses the asserted IDSEL line *and* the function number in the AD[10::8] signal lines to determine if it is being accessed. If IDSEL is asserted and the function corresponding to the function number in AD[10::8] has been implemented, the multi-function device asserts DEVSEL# to claim the transaction.

## ***INTERRUPTS***

The PCI bus defines four interrupt signal lines: INTA#, INTB#, INTC#, and INTD#. These signals are individually wired, or combined in various ways back to the interrupt controller by the system designer. It is important that all connectors have all four interrupt signals routed back to the PCI Interrupt Controller to accommodate any single- or multi-function application. The PCI Interrupt Controller must be capable of routing each individual interrupt to an IRQ, or must be able to support shared interrupts. The preferred design provides individual routing because some systems do not fully support shared interrupts.

The "Interrupt Pin" register, offset 3Dh, in the configuration header of the PCI device, defines which of the four PCI interrupt request pins (INTA#-INTD#) the device, or the function in the device, is wired to, as follows:

- 00h = the device or function does not use the interrupt pin
- 01h = INTA# used
- 02h = INTB# used
- 03h = INTC# used
- 04h = INTD# used

A single-function device must always wire the device's interrupt request signal to INTA# (Interrupt Pin register must be hardwired to 01h), and must never use INTB#, INTC#, or INTD#. A multi-function device, however, can implement one or more interrupt pins. There are two rules for multi-function chips and interrupts:

- Each function in the device can only be wired to one of the four interrupt pins (device must have at least as many functions as interrupt pins).
- If the device implements only one interrupt pin, it must be INTA#. If it implements two interrupt pins, it must use INTA# and INTB#, etc.

Functions in the multi-function device can share interrupt pins, e.g. an eight-function device can have all eight functions assigned to INTA#; or three could be assigned to INTA#, one to INTB#, two to INTC#, and two to INTD#; etc. This is called interrupt sharing, or interrupt chaining.

The PCI Specification states that "...the device driver may not make any assumptions about interrupt sharing. All PCI device drivers must be able to share

an interrupt (chaining) with any other logical device, including devices in the same multi-function package.” It is also important for operating system vendors to implement shared interrupts.

### **SYSTEM BIOS**

Systems designers must work with their BIOS vendors to make sure the system BIOS decodes functions 0-7 in the multi-function controller. The BIOS should do a configuration access to all functions if it detects that bit 7 of the “Header Type” register is set.

### **ARBITRATION**

There is a central PCI resource, usually part of the Host/PCI bridge, known as the central arbiter. The central arbiter connects to each bus master in the system via a separate pair of REQ#/GNT# signals. As mentioned earlier, a multi-function PCI device, like a single-function device, has a single REQ#/GNT# pair connected to it. A PCI bus master asserts its REQ# to tell the central arbiter that it wishes to be granted access to the bus. The central arbiter asserts the device’s GNT# signal to grant it access to the bus.

There is no defined arbitration scheme in the PCI specification; however, if more than one bus master is present in the system, the spec requires a fairness algorithm to avoid deadlocks, and to balance the different priorities of the various devices.

A multi-function device, since it represents one bus master to the central arbiter in the Host/PCI bridge, must implement an internal arbiter that is completely separate from the system’s central arbiter. The internal arbiter allows the different bus mastering functions within the chip to arbitrate among themselves for the privilege of arbitrating for PCI bus access. There may be multiple bus masters within the multi-function device. For example, in a SCSI/Ethernet multi-function device, there may be three separate internal bus masters: one for SCSI, one for Ethernet transmit, and one for Ethernet receive. It is the responsibility of the internal arbiter to arbitrate between these internal bus masters.

The PCI specification does not define internal device arbitration for multi-function devices. One method is to use priority levels. If there are three different channels in the device, for example, each can be assigned a programmable priority level. The

internal arbiter uses the priority levels to decide which internal function may arbitrate for access to the PCI bus. If two functions request access to the PCI bus simultaneously, the function with the higher arbitration priority level is granted access first. This priority scheme should be programmable so that the system can be tuned for different data requirements. A fairness algorithm should be implemented, similar to the central arbiter, to insure that no function gets starved for access to the PCI bus due to activity on other functions with higher arbitration priority levels.

### **SUMMARY**

The multi-function device, because of its single REQ#/GNT# pair, lends itself to either a motherboard or add-in card design. The device is a single bus master to the PCI host bridge, although internally it may be arbitrating for two, three, or more bus mastering functions. From the software perspective, the multi-function device contains a separate configuration space for each function.

There are a few considerations that must be taken into account when designing a multi-function device into a system; though as outlined above, the rules are simple and straightforward, and the benefits are numerous. The benefits of a multi-function PCI device include reducing the number of electrical loads and bus masters in the system, and freeing up valuable space on the motherboard and/or card slots for other features. As OEM customers continue to demand more and more features, PCI bus loading will become more of an issue. The multi-function device offers the OEM an opportunity to offer its customers the features they want and still meet the requirements of PCI.

### **REFERENCES**

- [1] PCI Special Interest Group, “PCI Local Bus Specification Revision 2.1,” June 1, 1995.
- [2] Edward Solari, George Willse, *PCI Hardware and Software*, Second Edition, San Diego: Annabooks, 1995.
- [3] Tom Shanley, Don Anderson, *PCI System Architecture*, Third Edition, New York: Addison-Wesley, 1995.



## **XVideo Family for PCI**

**XVideo, the developer's choice for high performance video, brings full-motion, 24-bit color, full resolution video to PCI-equipped Intel and PowerPC workstations. With a wide array of options to suit advanced users, XVideo supports multiple live video displays, JPEG compressed capture, storage, and broadcast. The Video Development Environment (API, sample applications, source code programs, etc.) is available to speed custom development and system integration. XVideo's fast JPEG hardware compression and decompression support desktop videoconferencing, multimedia authoring, distance learning, telemedicine, networked video distribution, and live video capture to disk. The VIO feature adds a second simultaneous live video display (composite, or S-VHS video format) from a second source and analog video output to monitor or VCR. Third-party software applications available. The PCI products interoperate with Parallax's Unix products for Sun and HP.**



# **3D***labs*

## **New Generation Silicon for 3D Graphics on PCI**

*Neil Trevett  
VP Marketing*

3Dlabs Inc.  
181 Metro Drive, Suite 520, San Jose CA 95110  
(408) 436 3456  
neil.trevett@3dlabs.com  
<http://www.3dlabs.com>



**3D***labs*

© 3Dlabs 1996 -PCI Spring 96 Page 1



## **3D Graphics on PCI is Hot!** *Topics Covered*

- Rapid market growth for 3D accelerator hardware
- Professional-class 3D
  - Breaking bottlenecks for more performance
- Consumer-class 3D
  - Making 3D pervasive
- Future directions for 3D

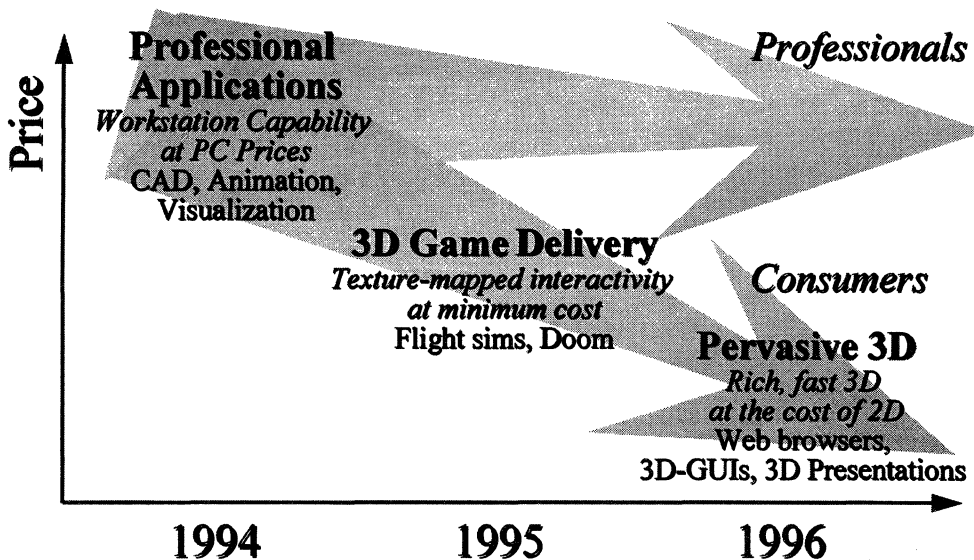


**3D***labs*

© 3Dlabs 1996 -PCI Spring 96 Page 2

## 3D Acceleration Market Dynamics

*Applications driving hardware sales*



- Two long-term market segments - Professional and Pervasive
- Games will be subsumed by Pervasive market

**3Dlabs**

© 3Dlabs 1996 -PCI Spring '96 Page 3

## Professional 3D Market Forces

*Driving the need for high-speed 3D*

- Applications
  - CAD - Pro/ENGINEER, MicroStation, SDRG/IDEAS
  - Multimedia Authoring - 3D Studio MAX, SoftImage, Lightwave
  - Scientific Visualization - AVS, Visual Numerics
  - Game Authoring - Gemini, Vigna, Multigen, Silicon Studio
  - Web Page Authoring - Microsoft, Netscape, SGI
  - Virtual Reality / Simulation - Sense8, Datapath, Gemini
- Pentium Pro
  - A true replacement for workstations / RISC CPUs
  - Needs high performance 3D to service applications

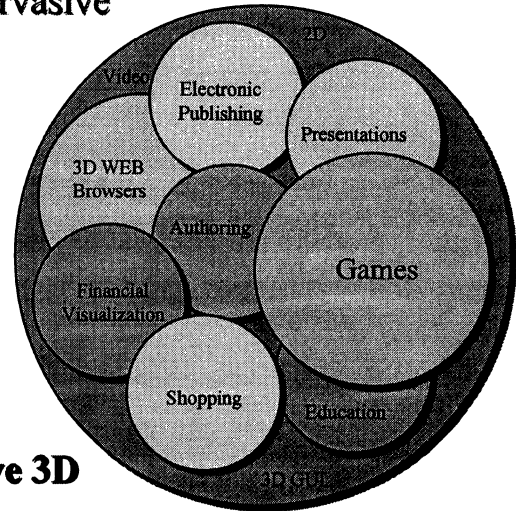
**3Dlabs**

© 3Dlabs 1996 -PCI Spring '96 Page 4

# Pervasive 3D Market Forces

## *What after Games?*

- Games are an important application of 3D acceleration, but 3D is becoming far more pervasive
- 3D will be used in everyday applications
  - You and I will use it!
- 'Pervasive 3D'
  - Subsumes 2D
  - Anything that manipulates pixels



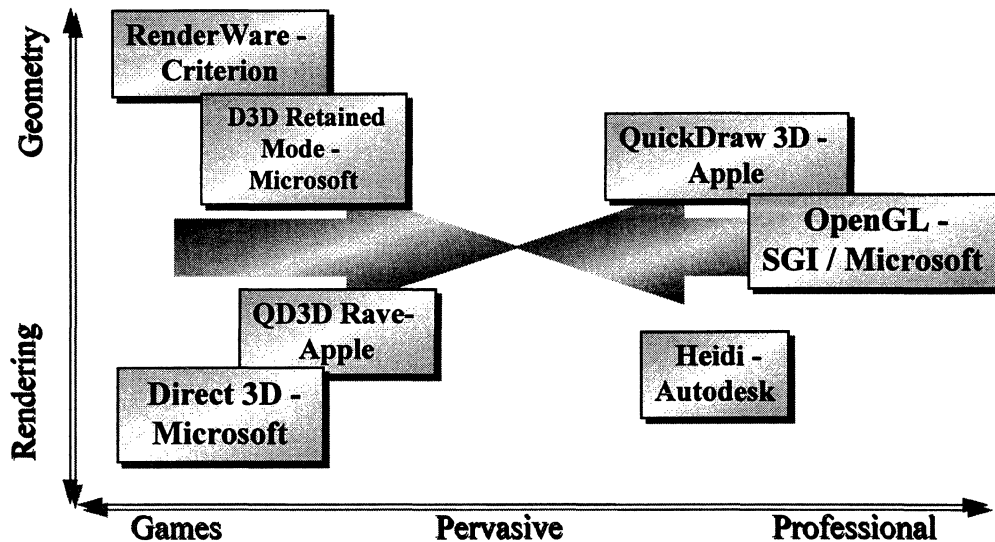
Pervasive 3D

**3Dlabs**

© 3Dlabs 1996 -PCI Spring 96 Page 5

# 3D API Landscape

## *The Rush to Service Pervasive Applications*



3Dlabs is developing drivers for the above APIs

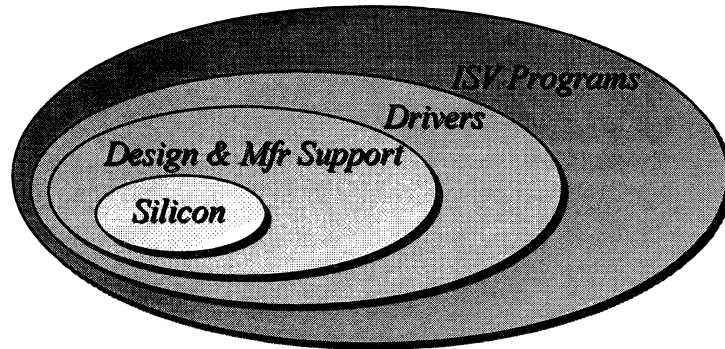
**3Dlabs**

© 3Dlabs 1996 -PCI Spring 96 Page 6

# 3Dlabs' Business

*A strongly focused company*

- Sell chips, technology and software for 3D graphics
  - Including 2D graphics and multimedia
- A 'Complete Partnership' approach to our customers
  - Not just silicon devices but chips, drivers and applications

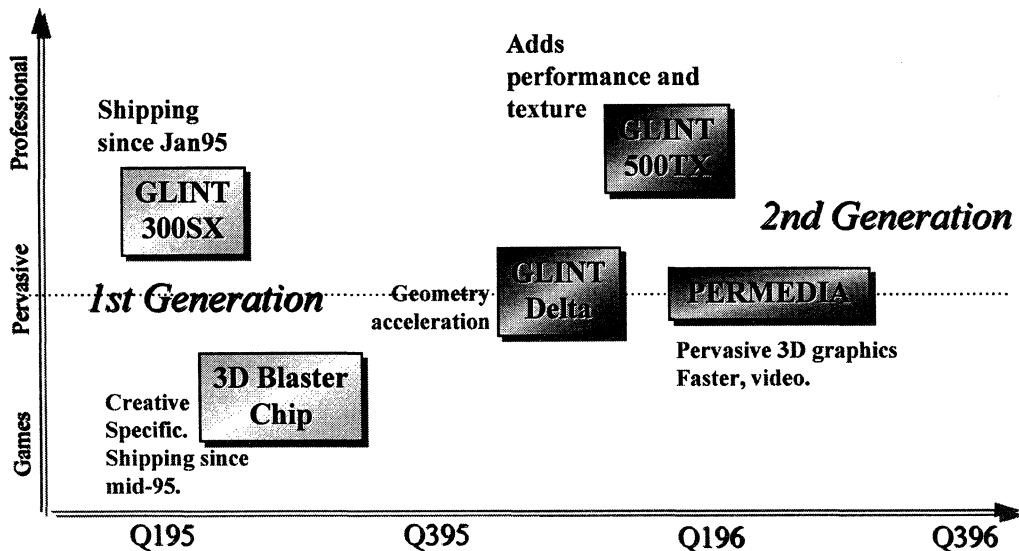


**3Dlabs**

© 3Dlabs 1996 - PCI Spring '96 Page 7

# 3Dlabs Merchant Chip Roadmap

*Extending leadership for 3D silicon*



**3Dlabs**

© 3Dlabs 1996 - PCI Spring '96 Page 8

# GLINT 500TX

*The next generation - half a million polygons/sec*

- 100% Pin Compatible with GLINT 300SX
- Uses single cycle EDO DRAM for localbuffer
  - Doubles pixel rendering rate to 25 Mpixels/sec
  - 500K polygons/sec - 32 bit shaded, Z buffered, 25 pixel
- Full Texture Mapping in silicon
  - Full perspective, filtered texture mapping
- Enhanced 2D Performance
  - Advanced optimized span operations
  - 2M vectors/sec
- Parallel GLINT 500TXs can drive a single framebuffer
  - Increased rasterization performance

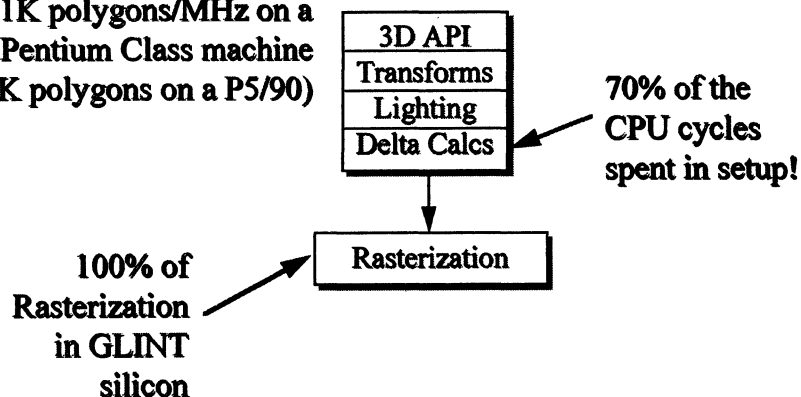
© 3Dlabs 1996 -PCI Spring 96 Page 9

**3Dlabs**

## But where's the bottleneck? *Geometry!*

- The fastest Pentium Pro cannot keep even a first generation GLINT saturated if running the geometry in software

1K polygons/MHz on a  
Pentium Class machine  
(90K polygons on a P5/90)



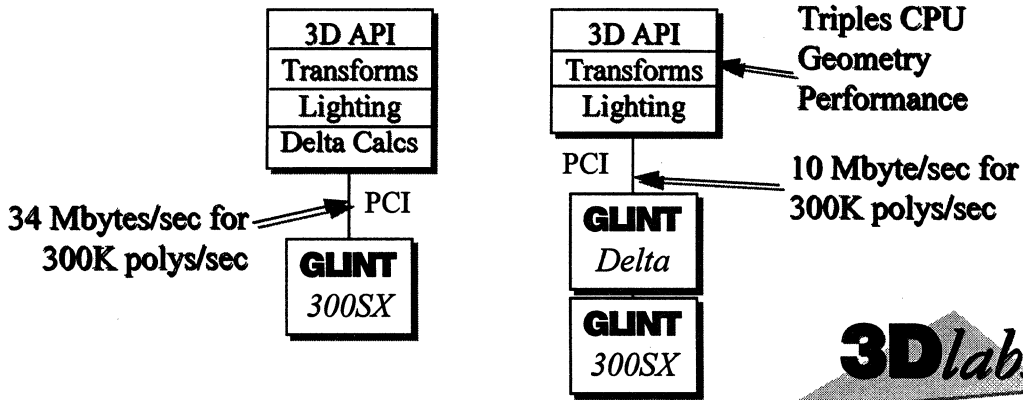
© 3Dlabs 1996 -PCI Spring 96 Page 10

**3Dlabs**

# GLINT Delta

## Breaking the Geometry Bottleneck

- GLINT Delta Chip - Hardwired 3D Pipeline Processing
  - 1M vertex/sec Vertex Setup Processor
  - Performs all delta calculations and floating point conversions
  - 100 MFlop floating point processor
  - Seamlessly integrates with GLINT software drivers
- Reduces PCI Bandwidth - just passing vertices - no slopes

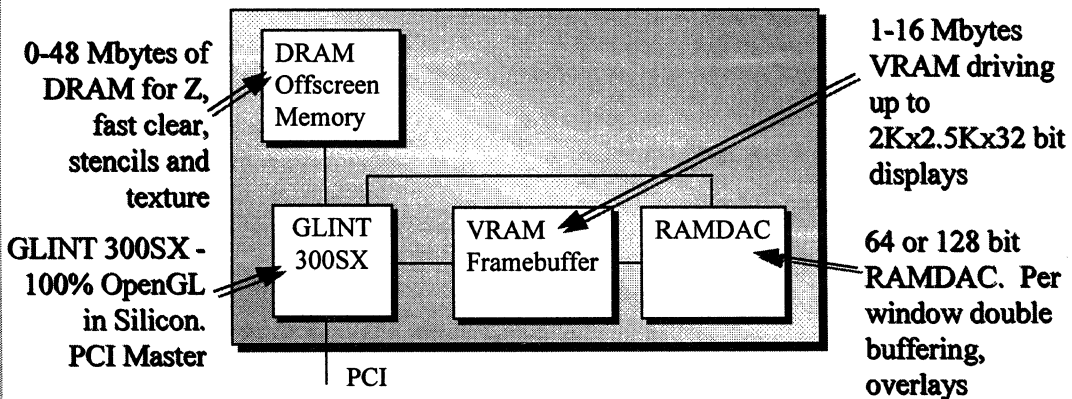


**3Dlabs**

© 3Dlabs 1996 -PCI Spring '96 Page 11

# First Generation GLINT Board

## The first Professional-class 3D accelerator for the PC



- GLINT connects directly to PCI, RAM and RAMDAC
- Two RAM banks for simultaneous, independent access
- Workstation class performance and functionality

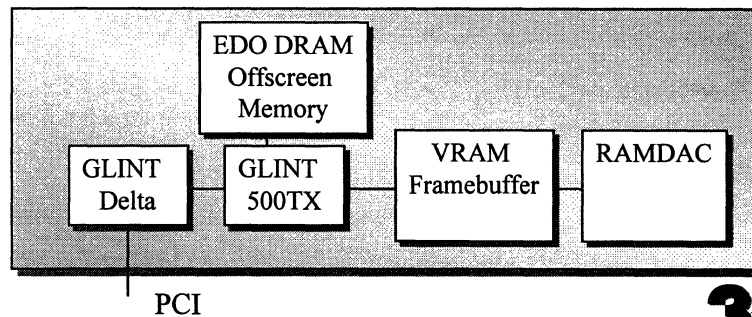
**3Dlabs**

© 3Dlabs 1996 -PCI Spring '96 Page 12

## 2nd Generation GLINT Boards

*A 5x speedup for the end-user*

- First Generation GLINT boards
  - Setup calculations bottleneck to about 90K polygons on P5/90
- Today's GLINT Boards
  - GLINT Delta breaks setup bottleneck
  - GLINT 500TX will deliver 500K polygons/sec
  - P6 - will drive geometry for 500K polygons/sec through Delta
- Jump from 90K to 500K polygons/sec in 12 months



**3Dlabs**

© 3Dlabs 1996 -PCI Spring 96 Page 13

## GLINT Delta

*Measured Performance with  
GLINT 300SX on Pentium Pro*

P6 @ 150 MHz, 1152x864, True Color, 75 Hz Screen Refresh

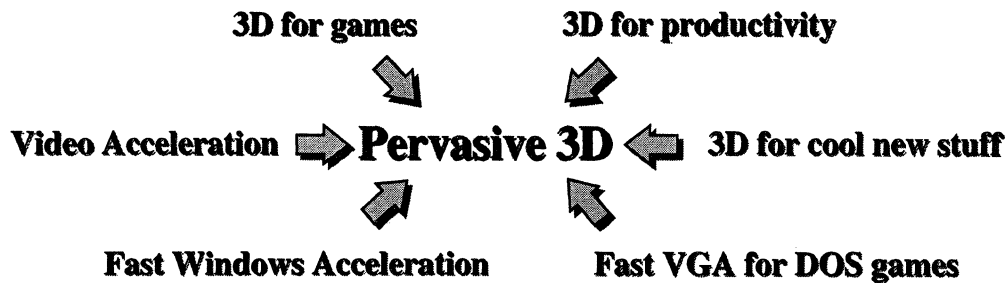
Tspeed3 V3.0 31 Jan 96	SX Only	SX+Delta	Speedup
Meshed Triangles (Z, Shaded) 50 Pixel per second	155,146	238,997	1.5
Meshed Triangles (Z, flat) 50 Pixel per second	205,870	321,247	1.6
Meshed Triangles (Z, Shaded) 25 Pixel per second	180,744	427,242	2.4
Meshed Triangles (Z, flat) 25 Pixel per second	232,398	573,212	2.5
Meshed Triangles (Z, Shaded) Small Triangles per second	187,454	599,762	3.2
Meshed Triangles (Z, flat) Small Triangles per second	249,629	586,527	2.3
Meshed Triangles (Z, Shaded) Single Pixel Triangles per second	187,454	600,476	3.2
Meshed Triangles (Z, flat) Single Pixel Triangles per second	249,629	586,527	2.3
Meshed Triangles (No Z, Shaded) 50 Pixel per second	182,048	277,016	1.5
Meshed Triangles (No Z, flat) 50 Pixel per second	223,155	365,412	1.6
Meshed Triangles (No Z, Shaded) 25 Pixel per second	199,290	514,781	2.6
Meshed Triangles (No Z, flat) 25 Pixel per second	272,531	585,847	2.1
Meshed Triangles (No Z, Shaded) Small Triangles per second	200,159	646,607	3.2
Meshed Triangles (No Z, flat) Small Triangles per second	271,068	586,527	2.2
Meshed Triangles (No Z, Shaded) Single Pixel Triangles per second	200,079	646,607	3.2
Meshed Triangles (No Z, flat) Single Pixel Triangles per second	271,214	586,527	2.2

**3Dlabs**

© 3Dlabs 1996 -PCI Spring 96 Page 14



## What's makes a Pervasive 3D Chip? *No compromises!*



- If you could get all this for the cost of 2D, no-one would need to buy graphics without 3D, and so 3D would become pervasive ...

**3D**labs

## **PERMEDIA™**

### *The first Pervasive 3D Graphics chip*

- No 2D compromises
  - More Windows performance than 64 bit VRAM controllers
  - Accelerated VGA for fast DOS games
  - Hardware video acceleration
- Fast 3D performance
  - Balanced performance for both textures and polygons
  - 500,000 textured polygons/second
  - Much more than just a games chip
  - Fully Compliant with D3D, OpenGL, Heidi, QD3D...
- Low cost
  - Uses unified SGRAM memory = low cost, high performance
  - A \$50 chip, selling on a \$250 board (2MBytes)
  - Inexpensive enough to be a games chip

**3D**labs

# PERMEDIA

## Performance Highlights

- 25 Million texture mapped pixels/sec
  - Bilinear-filtered with per pixel perspective correction
- 40 Frames/second textured frame rate
  - (640x400 screen, fully texture mapped, 2.5 depth complexity)
- 500 K texture mapped polygons/sec
  - Bilinear-filtered with per pixel perspective correction
- 1.6 GByte/sec 2D fill rate
  - Expecting 2x '968' class windows performance
- 30 fps video playback at 640x480
  - On-chip RGB-YUV conversion, scaling and bilinear filtering

© 3Dlabs 1996 -PCI Spring '96 Page 17

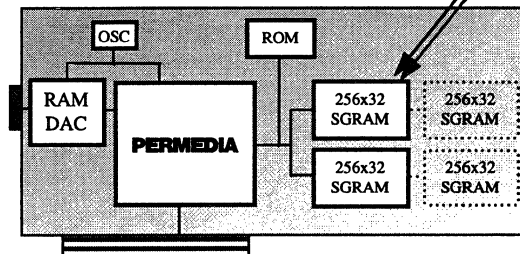
**3Dlabs**

# Board Design

## Low component count

- Single PERMEDIA Chip
  - plus SGRAM, RAMDAC, ROM and oscillator
- External interfaces
  - Glueless PCI Interface
  - High performance 64-bit SGRAM Interface
  - High speed pixel port
- PERMEDIA Packaging
  - BGA
  - 0.35 $\mu$
  - 3W at 3.3V

Typically 2MBytes,  
with optional upgrade  
to 4, 6 or 8MBytes



© 3Dlabs 1996 -PCI Spring '96 Page 18

**3Dlabs**

# PERMEDIA

## *Extensive Software Support*

- 3Dlabs' develops high quality, optimized drivers
  - Drivers provided free of charge to board customers
- Extensive API support
  - Microsoft's 3D APIs: Direct3D and D3D Retained Mode
  - Criterion's RenderWare
  - Productivity APIs - Heidi, OpenGL, QuickDraw 3D
- Creative has licensed CGL to 3Dlabs
  - CGL will be available on PERMEDIA-based boards
- Any Creative games title will run on PERMEDIA
  - Using CGL or other standard API
- Creative are pro-actively working with 3Dlabs to ensure PERMEDIA is the industry's leading silicon architecture

The logo for 3Dlabs, featuring the text "3Dlabs" in a bold, sans-serif font. The "3" is significantly larger than the "D" and "labs". The text is positioned above a stylized, shaded triangular shape that points downwards.

# Future 3D Hardware Trends

## *Exploring an unbounded opportunity space...*

- More Geometry in Hardwired Logic
  - Geometry processors that are pin-compatible with GLINT Delta
- Unified Memory
  - Using System Memory for texture, VESA's VUMA standard
- 3D Graphics on the Motherboard
  - High integration, unified memory
- A million polygon chip - in '96!
  - Including geometry!

The logo for 3Dlabs, featuring the text "3Dlabs" in a bold, sans-serif font. The "3" is significantly larger than the "D" and "labs". The text is positioned above a stylized, shaded triangular shape that points downwards.

# HIGH-SPEED DRAMS FOR PCI SYSTEMS

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

## **ABSTRACT**

A minimum design requirement for every PCI device is to understand the memory used in a PCI system. Although PCI technology is designed to separate the CPU, memory, and I/O buses in a PC, many of the chips in a PCI system must interface directly to memory. Designers of such chips today must deal with the changing landscape of DRAM memory selections available. A few years ago, page-mode and fast page-mode DRAMs were all that was available to designers. Today, high-bandwidth DRAMs (such as EDO, burst EDO, SDRAMs, RDRAMs, and a host of specialty graphics DRAMs, such as CDRAMs, 3DRAMs, MDRAMs, VRAMs, WRAMs, and SGRAMs) are available. Broadly speaking, these memory types can fall into three possible memory subsections: main memory, graphics memory, and a unified memory pool where both graphics and main memory exist. Each memory technology has advantages and disadvantages targeted at one or more of these memory pools. This paper explores the types of high bandwidth DRAMs and their characteristics targeted toward various memory subsystems found in PCI systems.

## **MAIN MEMORY ALTERNATIVES**

As page-mode DRAMS have evolved, two trends have become evident:

- Denser parts require wider I/O interfaces to maintain acceptable bandwidth.
- Memory granularity is minimized.

Until recently, only page-mode and fast page-mode (FPM) devices were available to designers. FPM devices allow processors to burst data at a maximum speed of X:3:3:3 (number of cycles for a cacheline burst) for a single bank; with bank interleaving, processors might be able to reach a speed of X:2:2:2 or X:2:3:2 (with a 66MHz Pentium bus). X is usually equal to 5 or 6, depending on the core technology used (i.e., -50ns or -60ns cores). -60ns cores are more standard today, but -50ns cores are expected to be standard in the near future.

Main memory performance is increased by decreasing the average number of wait states. Caches, write buffers, and a host of other techniques are used to mitigate DRAM accesses. However, sometimes it is necessary to access the DRAMs directly, and this access must be as short as possible.

For code fetches, the Intel Pentium processor bursts four 64-bit words at a time. The initial code fetch is for the instruction needed for the processor (not always at an offset of 0). The remaining fetches are for the remaining cache line entries. The initial latency is determined primarily by the RAS interval and any additional buffering or pipelining. The subsequent accesses are dominated by how fast a memory technology can return data specified by an "Intel order" during the burst. For the Pentium processor, the burst is always three additional clocks, and memory technologies take advantage of their highest speed page-mode transfers (or burst transfers) to satisfy this demand.

## **EDO DRAMS**

Extended Data Out (EDO) DRAMs are like conventional page-mode DRAMs with one exception: 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 difference, combined with a few other changes, allows EDO DRAMs to cycle faster in page mode, thereby 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. This also allows EDO parts to exist on stand x32 (or x36) memory SIMMs. EDO parts are one of the few main memory technologies that are applicable to both graphics and main memory.

For main memory applications, the denser 16Mbit parts are used in the x16 configuration, providing a minimum memory size of 8MBytes/bank. This minimum size is acceptable for the low-end Pentium systems in which Microsoft requires a minimum memory subsystem of 8MBytes for Windows 95 certification.

EDO devices increase performance for main memory applications to about X:2:2:2, which provides about a one- to three-percent performance boost for users. Although this performance increase is small, it has essentially no additional cost for users. Some people have even said that EDO is what DRAMs should have been all along.

### **BEDO DRAMs**

BEDO DRAMs further add burst sequence counters into the DRAMs to help decrease CAS cycle time.

Although faster core (-50) devices allow 1:1:1 burst operation, for the same core timing as that used in EDO DRAM today BEDO adds an additional wait state for the first access resulting in an (X+1):1:1:1 access. Although this is an overall improvement in access latency, the initial latency keeps the CPU stalling longer than EDO, although the entire burst is completed in two less cycles.

Overall, the jury is out on BEDO. Micron and several other companies are strongly supporting its use. Large users, such as Intel as well as the top DRAM companies, are working in different directions for high bandwidth memory solutions. The future success of BEDO is unclear, but it is currently a design alternative.

### **BEDO DRAMs**

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 several bus widths (x4, x8, and x16), mostly the wider parts on the 16Mbit DRAM density are suitable for main memory.

Although 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 memory controller designer must develop a new state machine in order to support SDRAMs.

Most SDRAM vendors offer parts that operate up to 100MHz, although the only interesting design point for Pentium systems today is synchronous with the 66MHz Pentium bus. Achieving even 66MHz operation on a board, using LVTTTL signaling will be very difficult because of board layout, clock/data trace routing, and skew issues. Memory expansion sockets further complicate the board design such that SDRAM DIMM modules must take into account loading of modules inserted and not inserted, as well as clock distribution.

Because of the pipelining in an SDRAM, it can achieve only (X+2):1:1:1 performance. This additional lead-off latency would negatively affect performance if the two banks were not used efficiently. By taking advantage of the two banks, some of the RAS precharge time can be hidden. If future addresses are known or can be estimated, some of the access time can be overlapped with the data transfer of the previous access. Systems implemented using such techniques are not available, but studies suggest that a two- to three-percent increase in overall performance can be gained by using these prefetching techniques.

### **RDRAMs**

Rambus DRAMs offer the highest bandwidth of the DRAM alternatives, transferring data at a burst rate of 533MBytes/s and now 600MBytes/s. 16Mbit densities are currently available with both x8 and x9 configurations. 64Mbit concurrent RDRAMs are under development with availability projected for early 1997. Suppliers include Hitachi, LG Semicon (Goldstar), NEC, Oki, Samsung and Toshiba. The RDRAMs are all pin-compatible and are offered in two surface-mounted packages: a vertical package for high-density packing on motherboards and a horizontal package for low-profile add-in cards.

The 64Mbit RDRAM has four banks which can be accessed concurrently. That is, while the contents of one bank are being read or written, another bank can be doing a RAS access cycle. The present 16Mbit RDRAM has a dual-bank architecture, whereas the 8Mbit version has a single bank. Instead of the conventional DRAM RAS/CAS interface, the Rambus Channel uses a multiplexed address/data bus. The

controller initializes each RDRAM on the Rambus Channel with a specific major address. When the controller is ready to perform an operation, it issues a packet on the Channel that requests the address of the data to be transferred. Only one RDRAM matches that address; the data is written or read at a burst rate of up to 600MHz.

Rambus DRAMs can transfer data much faster than conventional DRAMs because they use a new electrical interface referred to as Rambus Signaling Level (RSL). The Rambus Channel achieves its high speed with low-voltage signaling, terminated transmission line board layout, and precise clocking. The Rambus memory subsystem is a fully engineered solution using conventional printed circuit board layout and manufacturing processes.

Unlike other DRAM technologies, RDRAMs have been specifically designed to fit a wide range of applications, including main memory. The same RDRAM finds its way into graphics systems as well. Additional system solutions, such as sockets, expansion modules, and clock sources, are available from multiple suppliers.

The following table summarizes characteristics of DRAM technologies used for main memory applications.

**Table 1: 16MBytes Main Memory DRAM Comparisons**

	<b>EDO DRAM</b>	<b>BEDO</b>	<b>SDRAM</b>	<b>RDRAM</b>	<b>RDRAM</b>
Organization	1Mk x 16	1M x 16	1M x 16	2Mx8 or x9	8Mx8
Number of Chips Required	8	8	8	8	2
Bandwidth Per Chip	100 MBytes/s	132 MBytes/s	264 - 400 MBytes/s	533 - 600 MBytes/s	533 - 600 MBytes/s
Initial Lead-off Latency	X	X+1	X+2	X+4	X+2
Burst Latency	3:3:3 to 2:2:2	1:1:1	1:1:1	1:1:1	1:1:1
Package	40 SOJ or 40/44 TSOP	44/50-pin TSOP or 42-pin SOJ	44/50 TSOP	32 SVP or SHP	32 SVP or SHP
Pins Required on the Controller	95 - 110	80 - 160	68- 72 or 115-120	31	31

### **GRAPHICS DRAM ALTERNATIVES**

Until recently, there were only two DRAM choices: page-mode DRAM and video-RAM (VRAM). Today, the number choices has greatly increased. Page-mode DRAMs are being replaced with 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. MoSys DRAMs (MDRAMs) as well as a host of specialty DRAMs such as CDRAM, 3DRAM, etc. are available to the developer as well. But these parts are generally not applicable to cost sensitive PC designs as these parts are either expensive, or not widely available. The Rambus DRAM (RDRAM™) represents a revolutionary approach to increasing bandwidth. RDRAMs transfer data at 533MHz over a narrow, byte-wide bus referred to as the Rambus Channel.

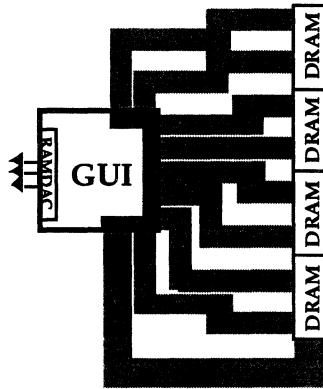
Whereas latency for burst fills is the common metric for main memory systems, graphics systems tend toward sustainable bandwidth. Because of the burst nature of so many of the graphics operations (display refresh, bit-blit, and so on), larger transfer sizes are normal.

### **EDO DRAMs**

EDO DRAMs for graphics systems are generally the same as those used in main memory designs except for one important difference. In order to provide sufficient bandwidth for graphics applications, the wide 16-bit I/O versions of these DRAMs are used in the 4Mbit generation parts. A 16Mbit EDO DRAM is not suitable

for graphics due to granularity and bandwidth issues. Although a 16Mbit DRAM provides sufficient storage for a 2MByte frame buffer, it does not provide sufficient bandwidth to meet associated display requirements. EDO parts are usually combined with a 64-bit bus by using four parts and require a minimum of 2MBytes of memory to make a 64-bit bus. EDO parts generally can be run with a CAS cycle time of up to 50MHz, providing a peak bandwidth of 400MBytes/s.

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 1MBytes 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.



*Single-Ported DRAM Approach*

Special versions of these parts are typically used in graphics systems, concentrating on reduced RAS cycle times. Core speed of -50 and even -40 are available, reducing the RAS overhead and keeping the usable bandwidth high, sometimes even approaching 80 or 90 percent of the peak bandwidth. Such a wide bus is accompanied by a "pin cost" on a controller—usually 90 to 100 pins.

Frame buffers using a 32-bit data bus to interface to single-ported DRAMs (EDO, SDRAM, SGRAM) use 52 to 55 signal pins plus 12 to 18 power and ground pins for a total of 64 to 73 pins.

### **BEDO Devices**

BEDO devices do not exist in densities that would be applicable to graphics.

### **SDRAMs and SGRAMs**

SDRAMs and a graphics specific SGRAM are alternatives for high-bandwidth graphics systems. These devices are applicable for graphics at the 4Mbit (x16) and 8Mbit SGRAM (x32) densities. Although 16Mbit SDRAMs are applicable to main memory applications, the 16-bit wide bus is too narrow to support the display refresh performance requirements of the majority of important display sizes.

SGRAMs, which are based on SDRAMs, are offered in the 8Mbit density. They use a x32 interface 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 on every other clock cycle). They also have a single-color register, so color expansion requires two passes for a font. Block write increases bandwidth 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.

Data sheets have become available for SGRAMs, with some vendors showing 100MHz bin split parts. Because of board layout, clock/data trace, and skew issues, achieving 100MHz operation on a board using

LVTTL signaling will be very difficult. Memory expansion sockets further complicate the board design; SGRAM configurations supporting expandable frame buffers are not expected to achieve the component's specified 100MHz operation in a system environment.

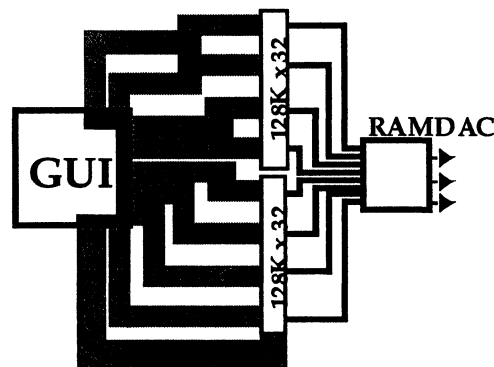
A few controllers using SGRAM have been announced but are not yet shipping. Highest speed claims are currently for 83MHz operation. Wide bus versions of the interface still have a significant pin penalty (usually higher than EDO) due to the higher frequency operation requiring more ground pins to minimize ground bounce. For a 32-bit bus running at 66MHz, the bandwidth is just 267MBytes/s. Running at 83MHz, it would be 333MBytes/s; at 100MHz, it would yield burst speeds of 400MBytes/s. Doubling the bus width to 64-bits would double these numbers.

### VRAMs and WRAMs

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 their dual-port design and other features, VRAMs traditionally have been significantly more expensive than single-ported DRAMs and, therefore, 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 to 50 percent 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 move capability—positioning the part for high-performance add-in cards. The WRAM is currently single-sourced, which is a concern to graphics card manufacturers.

In general, dual-ported memories are being bypassed for cost-sensitive applications, since the fixed bandwidth partitioning of the two buses offers no advantage at low-resolutions and the serial bus cannot be used for extra drawing bandwidth. Also, the trend for cost-sensitive, high-volume applications encourages chip designers to integrate the RAMDAC, causing the pins of the VRAM serial port to go back to the GUI chip, further increasing the pins and cost.



*Dual-Ported DRAM Approach*

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.
- The DRAM 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 and could force the controller into a larger, more expensive package.

### MDRAMs

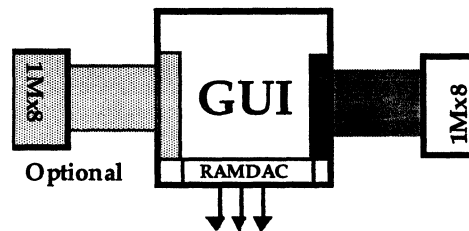
A relatively new entrant in this crowded field of graphics specialty memories is MoSys. They have copied the Rambus approach by providing multiplexed address data on a data bus that transfers data on both edges of the clock. Their data bus width is 16 bits. Unlike all other DRAMs, MDRAMs differentiate by providing many banks, as many as 96 banks, according to their data sheet. Because of the granularity of the banks, MoSys claims to be able to offer parts in memory sizes other than powers of two (such as .75MByte, 1.125MBytes, and 2.3MBytes). There are a total of 26 active signal pins and 16 power and ground pins on each MDRAM. MDRAMs are offered in either an 86-pin PLCC package or a 160-pin PQFP.

Although the granularity offers specific sizing, it also requires eight standard parts. Fitting display size to the exact storage requirements also results in no off-screen storage. In all versions of hardware requirements necessary for Windows 95 certification (and beyond), Microsoft requires 1MByte or larger display buffers. MoSys also claims to be able to run up to 166MHz, with an effective bandwidth of 666MBytes/s, using standard LVCMOS signaling.

### RDRAMs

Rambus DRAMs offer the highest speed DRAM alternatives, transferring data at a burst rate of 533 or 600 MHz. 8Mbit x8 and 16Mbit x8 and x9 densities are available targeting graphics applications. All RDRAMs are pin-compatible and are offered in two surface-mounted packages: a vertical package for high-density packing on motherboards and a horizontal package for low-profile add-in cards.

The 16Mbit RDRAM has a dual-bank architecture, whereas the 8Mbit version has a single bank. Enhancements to existing RDRAMs are being released which substantially reduce latency and increase operating frequency to 600MHz. These parts will be available this year and the additional bandwidth can be used in graphics applications for higher-resolution, deeper frame buffers, higher refresh rate monitors, better performance or additional functionality.



*Rambus Approach*

Rambus technology is well suited for graphics due to its high bandwidth and the flexibility of product offerings that it allows. A single 600 MBytes/s Rambus Channel supports sufficient bandwidth for display resolutions up to 1280 x 1024 x 24bpp (or even 1600 x 1200 x 16bpp). A single Rambus Channel is able to cover most PC display market requirements and up to 4MByte display buffers. GUI controllers can also support two Rambus Channels providing up to 1.2 gigabyte-per-second in bandwidth for higher resolutions or higher performance systems. The 8Mbit and 16Mbit RDRAMs allow the lowest-cost, single-component 1MByte or 2MByte frame buffers. The 8Mbit and 16Mbit RDRAMs are pin-compatible, allowing the manufacturer to support multiple frame buffer sizes with one board layout. By incorporating a low-profile RSocket™ on the Rambus Channel, memory can be expanded at product build time by the dealer or even by the consumer with low-cost memory modules (RModules™).

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 of the pins for 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 into the controller, such as support for video interfaces, feature connectors, or 3D graphics support. The Rambus-based frame buffer allows the lowest cost controllers and lowest cost support for expanded feature sets.

Unlike the SGRAMs or dual-ported DRAMs, RDRAMs have been specifically designed to fit a wide range of applications, including, but not limited to, graphics. The advantage to graphics designers is that broadly used memories ramp in volume and descend price curves faster than components targeted for specific application segments. A comparison summary of the alternatives is shown in Table 2.

Due to its compact design, Rambus technology helps to reduce board space. Most PC graphics frame buffers can be implemented with a single 8Mbit or 16Mbit RDRAM component. With the possible exception of SI/SO), 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 overall cost of the 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. In addition, minimizing the "footprint" on the motherboard increases the feasibility of putting the graphics controller directly on the motherboard.

**Table 2: 2MByte Frame Buffer Comparisons**

	<b>EDO DRAM</b>	<b>4Mbit VRAM</b>	<b>WRAM</b>	<b>SDRAM</b>	<b>SGRAM</b>	<b>MDRAM</b>	<b>RDRAM</b>
Organization	256K x 16	256K x 16	256K x 32	256K x 16	256K x 32	197K x 32	2Mx8 or x9
Number of Chips Required	4	4	2	4	2	2	1
Peak Bandwidth Per Chip	100 MBytes/s	50 MBytes/s	160 MBytes/s	132 - 200 MBytes/s	264 - 400 MBytes/s	400 - 666 MBytes/s	533 - 600 MBytes/s
Relative Cost	1.00	1.9	1.5	1.1	1.3-1.6	?	1.05
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	1.0 sq. in.	0.1 sq. in vertical 0.5 sq. in horiz.
Package	40 SOJ or 40/44 TSOP	64 ZIP	120 PQFP	44/50 TSOP	100 TQFP	68 pin PLCC or 128 pin PQFP	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	70 - 80	31

### **BEYOND MAIN MEMORY AND GRAPHICS**

As DRAM densities continue to increase, fewer DRAMs are needed for a given memory subsystems. Already in graphics subsystems, a single DRAM can satisfy most of the graphics operations for a PC. Inevitably, main memory will be forced to one or two chips, which will spur designers to try to integrate graphics and main memory into the same memory space. These types of unified memory subsystems are the likely future for PC systems.

Already, game machines are showing the way. Nintendo is introducing its revolutionary Nintendo 64 game system with a unified memory subsystem. Consisting of a single Rambus channel, all graphics data, display refresh, executable code, and data are contained in one memory space. Because of Nintendo's use of RDRAMs, they were able to accomplish this in an extremely low-cost consumer product.

### **CONCLUSIONS**

1996 will be the first year any of the new high-bandwidth DRAM memory type reach any significant volume. Several graphics companies (SGI, Cirrus Logic, and Chromatic) all will be shipping systems or

chips that use the Rambus interface. An important game platform, the Nintendo 64, is using RDRAM technology as a unified memory subsystem. Reaching unparalleled graphics capability in an \$250 base system, this machine promises to raise the bar on expectations from 3D subsystems and cost.

### ***BIOGRAPHY***

Billy Garrett is the manager of graphics development at Rambus Inc. While at Rambus Inc. he has worked on the development of Rambus technology for high volume personal, portable, and multimedia systems. Mr. Garrett holds both a BSEE ('82) and an 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.

# Using PCI Interface in Routers

*Aamer Mahmood, Cisco Systems*

For its existing and next generation routers, Cisco has chosen network interface architecture that uses PCI bus as a common interface between various network media adapters, called Port Adapters, and the rest of the system. This allows leveraging of industry standard chips and also makes it possible to use the Port Adapters across multiple platforms. The Port Adapters interface to the network media (ethernet, token ring, fddi, etc.) on one end and connect to the rest of the system at the other end using PCI bus. The adapters are compliant to PCI electrical specifications. Mechanically they are 6.5 x 5.5 (roughly), double-sided boards that use 200-pin AMP connector that supports hot swap across the PCI bus. The system uses multiple levels of PCI buses. There can be many Port Adapters in a system and each Port Adapter can in turn support 1–8 PCI agents. External Arbiter on the Port Adapter is used for arbitrating bus between the host system (to which Port Adapter is connected) and multiple agents on the PCI bus on the Adapter. All the Adapters have master capability and transfer data from/to network interface using 16-byte (or larger) bursts. All Adapters are required to support scatter/gather DMA with no alignment restrictions. As most of systems are big-endian, it is highly desirable that adapters support both little and big endian nodes. As latency is of concern, all Adapters are required to have big enough fifos to support the network interface at line rate. Since there are multiple Port adapters in a system, PCI bus utilization by each PCI interface is of primary importance.



**ESSENTIAL**  
**Communications**

## Serial-HIPPI Network Interfaces Using the RoadRunnerPCI ASIC

*Michael McGowen*  
Chief Technical Officer

---

Copyright 1995, Essential Communications.  
All rights reserved under International and Pan-American  
Copyright Conventions.

*World Fast Networking* is a trademark of Essential Communications.  
All other trademarks and copyrights belong to their respective companies.

Material in this overview subject to change without notice. Essential  
Communications assumes no liability for errors or inaccuracies in this document.

Essential Communications  
4374 Alexander Blvd., Suite T  
Albuquerque, New Mexico 87107  
U.S.A.

Tel. 1-505-344-0080  
Fax. 1-505-344-0408

info@esscom.com  
<http://www.esscom.com>

## Introduction

In addition to providing a full line of HIPPI switches, gateways and related products, Essential Communications has developed a number of network interface cards supporting different bus interfaces, thus providing Serial-HIPPI networking for a wide range of workstations and personal computers. The heart of these cards is a custom ASIC known as the "RoadRunnerPCI."

RoadRunnerPCI Serial-HIPPI adapters are designed to provide flexibility and high performance, and feature:

- PCI (Version 2.1) host interface,
- 800 Mbit/sec HIPPI data rates in both directions simultaneously,
- on-board intelligence that reduces requirements on the host processor,
- event-driven rather than interrupt-driven processing,
- unique ring-buffer approach to data movement,
- low latency,
- TCP/IP checksum assistance,
- Endian data conversion.

Conceptually, the RoadRunnerPCI Serial-HIPPI adapter utilizes a "building blocks" architecture: in addition to its internal processor, the various interfaces provided on the card are virtually independent objects that can be easily replaced or upgraded. This allows, for example, creation of a new card that maintains the same host interaction but uses a different bus interface. Or, the on-board processor could be replaced with one that is more powerful. Even the HIPPI interfaces can be replaced with other network interfaces.

## What Is Serial-HIPPI?

High Performance Parallel Interface, or "HIPPI," is today's industry standard for high-bandwidth networking in both system-to-system and system-to-peripheral environments. Standardized by the American National Standards Institute (ANSI), HIPPI has been widely adopted by research, higher-education and engineering organizations worldwide. This transition is occurring because other standard networks, previously thought of as high bandwidth (e.g., FIDDI, ATM-OC3), cannot keep up with today's systems.

With built-in features for high-bandwidth network switching, HIPPI defines multiple point-to-point channels between CPUs, and from CPUs to storage systems, displays and other peripherals. HIPPI provides a bandwidth of 800 megabits (100 megabytes) per second over a distance of 50 meters on copper cables. Serial-HIPPI extends this distance to one kilometer on multi-mode, and 10 kilometers on single-mode fiber optic cables, while providing two channels, one each for simultaneous transmit and receive.

HIPPI networking has long been used in production supercomputing environments for peripheral connectivity, clustering and high-speed LANs. Workstation-based HIPPI networks were difficult to implement because of the bulky 50-pair copper cables that were required. Now, however, with Serial-HIPPI available on network interface cards, large-scale HIPPI LANs are not only viable, but easy to implement.

## Basic NIC Architecture

In addition to this PCI card and a PCI mezzanine card, the NIC family includes cards designed for use with Silicon Graphics' GIO-64, Sun's SBus and IBM's MicroChannel architectures. The latter cards feature a bridge between the particular bus interface and the RoadRunnerPCI (the bridge is not needed on the PCI-bus cards); this hardware bridge is transparent to the RoadRunnerPCI's host interaction software. Thus, the complete range of host software drivers and utilities developed by Essential is available for all of the network interface cards.

The basic NIC hardware consists of a Serial-HIPPI fiber transceiver, two HP GLINK chips, the RoadRunnerPCI ASIC, and a memory chip.

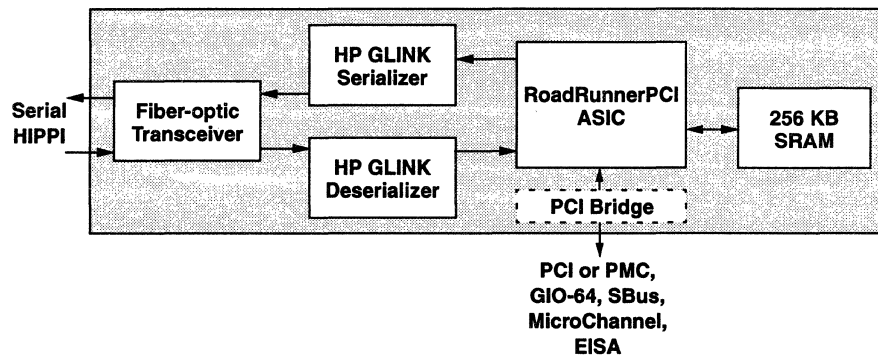


Figure 1. Network interface card layout.

The RoadRunnerPCI interfaces directly with the PCI bus and contains an on-board 32-bit RISC processor, DMA engine, PCI registers, and local registers. The internal CPU manages the independent DMA channels between the host and the on-board memory (256 KB of SRAM), and between this local memory and the HP GLINK HIPPI serializer chips. Conversion between serialized and optical data is via a Methode optical transceiver. The NIC operates on a single +5V source.

PCI mezzanine cards (PMCs) are used where slim, parallel board mounting is necessary, such as in a single-board computer host, or for media interface in systems with no PCI bus. This PMC HIPPI card, with the Essential driver suite, is intended for use in embedded systems and previous-generation workstations with bus architectures such as HP Precision or VME.

Essential is the first vendor to provide completely integrated Serial-HIPPI network interface cards—all previous HIPPI NICs were parallel, copper versions. The RoadRunnerPCI ASIC also dramatically reduces the NIC's complexity and number of components; thus these cards are much lower in cost and directly competitive with the latest ATM and Fibre Channel offerings. The Essential driver supports TCP/IP HIPPI ARP (RFC 1374), the HIPPI Network Forum API, and IPI-3 diagnostics and performance-monitoring utilities.

Supported bus platforms are PCI Version 2.1, PCI Mezzanine card (PMC), GIO-64, SBus, MicroChannel, and EISA. Supported operating systems are Digital UNIX, Windows NT, Novell Netware, SGI's IRIX, Sun's Solaris, IBM's AIX, HP-UX, Wind River's VX Works, and FreeBSD.

## 1.0 Features

The RoadRunnerPCI Serial-HIPPI NIC is a programmable device that includes two network interfaces: data transmission is via a single-wide Source port, and reception occurs over a single-wide Destination port—these are separate, 800 Mbit/sec. optical-fiber channels. The two ports are usually used together to form a shared full-duplex device, but the channels can be used independently as two non-shared private devices. This means that the network interface card can separately support an application that just sends and another application that just receives, as well as applications that both send and receive.

The RoadRunnerPCI's CPU performs all necessary HIPPI-PH physical-layer processing. The host processor and software simply provides packets to be sent and accepts received packets—all HIPPI operations are transparent to the host. Special provisions are included for dealing with unlimited size packets, as well as for concatenating several packets within the same connection.

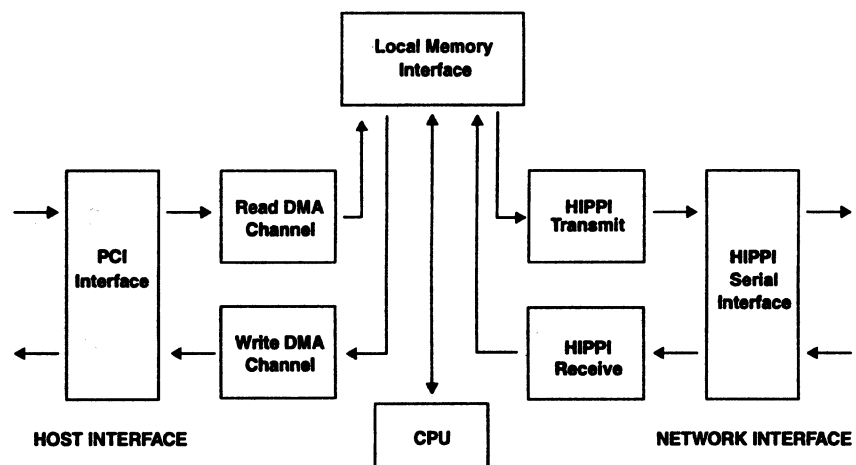


Figure 2. The RoadRunnerPCI ASIC.

The PCI Local Bus Specification Version 2.1 supports a 64-bit data path in addition to the standard 32-bit data path. The RoadRunnerPCI Serial-HIPPI adapter provides a 32-bit interface. However, the RoadRunnerPCI utilizes a 64-bit datapath internally, so future versions of the RoadRunnerPCI Serial-HIPPI adapter could easily support a 64-bit PCI host interface.

### 1.1 Internal Processor

The primary interaction between host and adapter is between two processors—the host is considered as one, and the CPU on the RoadRunnerPCI is the other. This approach allows the addition of new or unique features via simple software upgrades. The RoadRunnerPCI processor can be set up to accommodate different buffer descriptor formats, providing the ability to adapt to the communication method which best suits the host.



Processor execution is according to an “event” model rather than an “interrupt” model, meaning that the frequency of host interruptions is greatly reduced, which contributes to higher performance execution. A small instruction cache is provided so the processor does not use any of the NIC’s local memory bandwidth.

The NIC architecture uses card-based, master-mode DMA (direct memory access) to move data efficiently between host memory and the NIC memory over the PCI bus. The RoadRunnerPCI processor performs all tasks required to operate the fiber HIPPI interfaces and control the DMA channels. The processor directs the operation of the adapter, based on downloaded configuration parameters; it also keeps processing and interface statistics, which are accessible to the host.

## 1.2 DMA Channels

The RoadRunnerPCI Serial-HIPPI adapter provides DMA functions for moving data to and from host memory, eliminating the need for the host to perform these operations. Two independent DMA channels are provided—one is used exclusively for host memory reads while the other is used exclusively for host memory writes. These channels provide the primary means by which the PCI accesses host memory (mailboxes are also implemented, as discussed in a following section), and they are used for the transfer of control information, as well as network data, all of which is managed by the internal processor.

## 1.3 Buffer Rings

Data is moved by means of host-provided memory buffers. Each buffer is identified by a *descriptor* which includes a pointer to the buffer location, the length of the buffer, and control flags that indicate the content of the buffer.

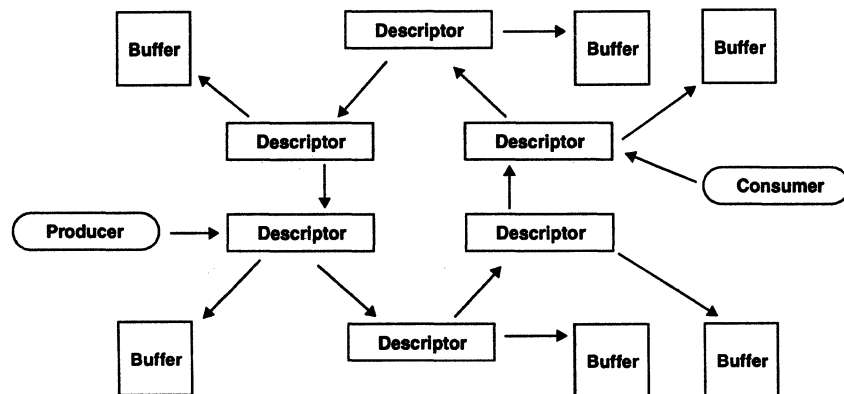


Figure 3. A buffer-descriptor ring.

All bytes in a buffer are physically contiguous when viewed from both the host and the NIC. (Buffers are frequently one virtual page long.) Note that this buffer-descriptor approach supports the “scatter/gather” method of memory usage. Removing the responsibility for buffer coordination from the host processor increases its efficiency and overall system performance.

Buffer descriptors are collected into fixed-sized rings for processing (see the previous figure). In addition to pointing a buffer in host memory, each descriptor points to the next descriptor, and the last descriptor points to the first.

Internally, a ring is an array of descriptors. Processing proceeds from descriptor to descriptor around the ring. The agent that adds buffers to the ring is called the “Producer,” and the agent that removes buffers is called the “Consumer.” The Consumer chases the Producer, as shown in Figure 4; when the Producer and the Consumer reference the same descriptor, the ring is empty. (The Producer is not allowed to “catch” the Consumer.)

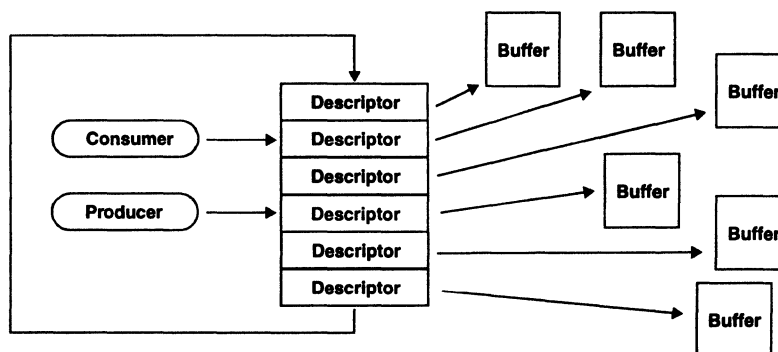


Figure 4. Descriptor-ring processing.

As described earlier, data is actually transferred between the host and the NIC by means of master-mode DMA channels, controlled by the RoadRunnerPCI.

### 1.3.1 Ring Management

The RoadRunnerPCI ASIC supports four types of rings: Receive, Send, Event, and Command. All data processed by the NIC pass through a data ring—there is one Send ring and one or more Receive Rings (when operating in HIPPI-PH mode, there is only one Receive Ring). Receive Rings handle incoming data and the Send Ring handles outgoing data. Receive Rings are created and destroyed as device interfaces are opened and closed.

The two control rings manage the operation of the RoadRunnerPCI Serial-HIPPI adapter. The Command Ring handles commands from the host software that the NIC will process, and the Event Ring contains the results of NIC processing. Typical commands include notification that there is more data to send, and a notification to the NIC that there are more empty buffers for a Receive Ring. Typical Events include “Packet has been sent” and “Packet has arrived.”

Data buffering is managed through the ring-buffering mechanism described earlier. Received packets are multiplexed by the NIC based on the HIPPI-FP ULP field and placed in the appropriate Receive Ring. The frequency of host interrupts is reduced by the RoadRunnerPCI placing Event notifications in an Event Ring—interrupts are then generated based on conditions in the Event Ring instead of for each Event.

## 1.4 Data Synchronization and Buffer Alignment

The NIC also incorporates local Send and Receive Rings that are not visible to the host (see Figure 5). These rings are used for buffering and proper synchronization of data as it passes between the host and the HIPPI interfaces.

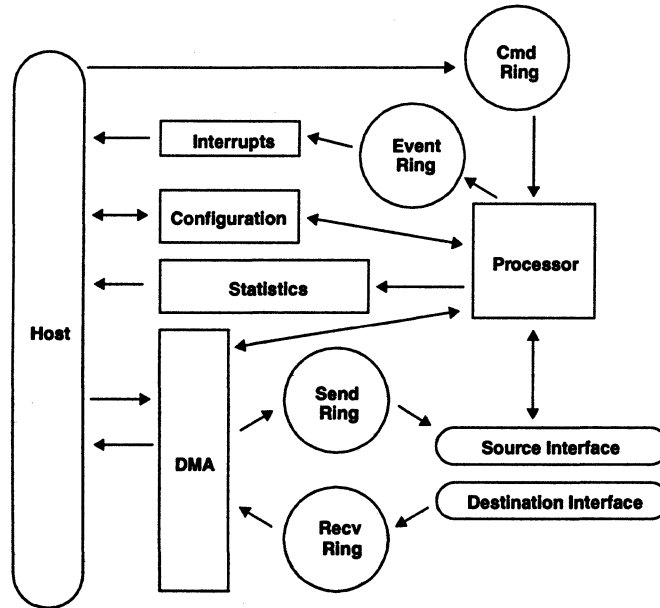


Figure 5. The RoadRunnerPCI Serial-HIPPI adapter model.

This approach supports automatic handling of misaligned buffers, which means that data can be transferred between any two buffers, regardless of the alignment of either. In addition, this feature can eliminate the need for the host to make copies of the data, providing a major improvement in performance.

### 1.4.1 Buffer Alignment

Not all DMA transfers will be nicely aligned to 32-bit boundaries; in fact, the data could be on any byte boundary. Traditionally, most adapters have placed the burden on the host to align the buffers prior to data transfer. This is time consuming and degrades system performance. The RoadRunnerPCI alleviates the problem by providing the byte steering logic necessary to compensate for misaligned buffers.

Data from any host byte address can be aligned to any byte offset of the RoadRunner's internal memory. This flexible scheme allows multiple odd-length, odd-boundary buffers to be directly concatenated into the proper 32-bit HIPPI words. This process also aids in the generation of the internal TCP/IP checksums.

## 1.5 Programmable Configuration and Operating Firmware

An on-board EEPROM is used for storage of important configuration and manufacturing information. The internal processor initially executes from this

EEPROM, loading all of the internal PCI registers. (Note that the present register configuration will remain on the adapter even if it is moved between machines.)

Use of the EEPROM allows vital adapter information to be accessed and updated via software whenever necessary—both the host and local processor can read and write this information. The EEPROM also can store non-PCI information, such as addresses, manufacturing data, diagnostic results, etc.

### 1.5.1 Operating Firmware

The software running on the internal processor is referred to as “firmware” to prevent confusion with the host driver. The firmware image runs from on-board SRAM, which is loaded during initialization, either from host memory or from the on-board EEPROM. The host also can store firmware in the EEPROM. Thus, software conflicts (e.g., incompatible or outdated versions) between host driver and adapter are reduced. In addition, installation of new software is simplified—PROM swapping or use of a special firmware-loading utility is not required.

Firmware images include operating images and diagnostic images. Each mode is independent of the other modes. There is an operating image that supports HIPPI-FP operation, with checksum extensions, and a separate image that supports HIPPI-PH processing. The NIC can be configured either as a shared device supporting HIPPI-FP, or as a dedicated point-to-point, private-protocol device supporting HIPPI-PH.

The shared model is suitable for network operation using IP concurrently with applications using private protocols. In the shared mode, a single connection can pass a limited amount of data in one or more packets. The NIC is configured for maximum number of bytes that can be passed over a single connection. In all cases the NIC handles the HIPPI-PH state transitions and all request, confirm, indicate and response primitives.

### 1.5.2 Checksums

In addition to support for TCP/IP and UDP/IP checksum generation and validation during HIPPI-FP operation, a TCP/IP-style checksum is calculated on all data transferred through the RoadRunnerPCI's internal data rings. This can improve the performance of a host implementation that supports hardware checksum assist.

## 1.6 Error Handling

There are two possible error types defined by the PCI specification, parity errors and system errors. The RoadRunnerPCI controller supports both of these error reporting mechanisms.

Parity errors are checked across the data bus by the receiving agent on the PCI bus during transactions. System errors are checked across the address bus, and across the data bus during a Special Cycle command. Any interface on the PCI bus can indicate a system error. Certain error conditions cause events to be sent directly to the ASIC internal processor. This allows flexibility in the implementation of an error-recovery mechanism.

## 1.7 Memory Arbitration

There are six entities that arbitrate for the use of the local SRAM. A priority scheme between these six requesters ensures no requester can cause a loss or corruption of the dataflow.

The two HIPPI network interfaces have the highest priority so they never underrun or overrun. The HIPPI Transmit interface is guaranteed access to the local memory on the clock cycle after it makes a request; the HIPPI Receive interface is guaranteed memory access within two clock cycles of making a request. The Transmit or Receive hardware then ensures that there is at least one, and sometimes two, clock cycles between memory accesses—this enables other requesters access to the memory.

Host access to the RoadRunnerPCI also must occur in a timely fashion, and is next in priority. Depending on the level of HIPPI activity, a host access request is serviced between one and five internal clock cycles.

The internal processor is considered lowest priority as long as any data transfers are in progress. Since the primary job of the processor is to keep the data moving, it's job is most critical prior to data movement beginning, at which time there will be plenty of bandwidth for the processor. The processor is never starved for bandwidth, since it has an instruction cache, as well as a mechanism that advances its priority if its request has been waiting too long.

## 1.8 Mailboxes

A common technique for communication between a host processor and an adapter is use of "mailboxes." Typically these are locations that are written to by one processor, causing an interrupt to the other processor. The value written may or may not have any significance, and each processor is allowed to read the mailbox only once before it is cleared by the hardware.

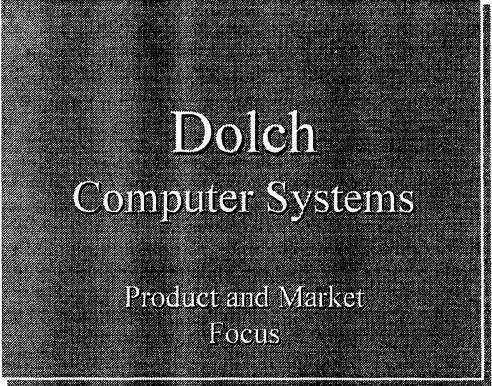
In the RoadRunnerPCI, the main mechanism for communications between the host and the RoadRunnerPCI adapter is via buffer descriptors. Use of host buffer descriptors is more efficient, allowing larger amounts of data to be passed without requiring the host to access the PCI bus or process interrupts. However, generalized mailboxes also have been implemented to provide greater flexibility in the interaction between the host and the adapter. The value in the mailbox is significant in this mailbox scheme. Also, mailboxes can vary in size, and can be read as many times as necessary without the contents being erased.

## 1.9 Endian Conversion

There are two basic formats for storing data in memory: "Little Endian" and "Big Endian." The PCI Local Bus Specification (Version 2.1) prescribes a Little-Endian format for PCI buses; however, not all hosts accept data presented in this format. Therefore, to facilitate communication between hosts and adapters with Big Endian designs, the RoadRunnerPCI can perform Little-Endian to Big-Endian byte swaps, translating between the two formats. This flexibility means the RoadRunnerPCI also can be the basis for a non-PCI interface where data is expected in Big-Endian format, and it enhances the performance of PCI bridges to other bus systems such as GIO-64, SBus and MicroChannel.

# Dolch Computer Systems

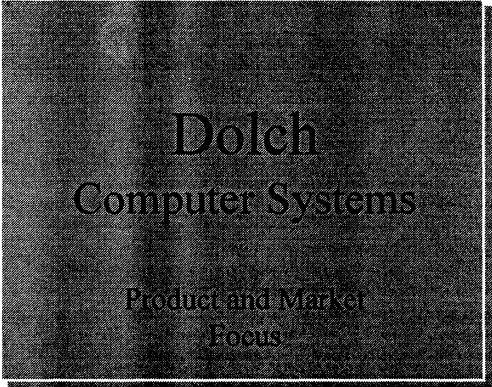
**Introduction**



Dolch  
Computer Systems

Product and Market  
Focus

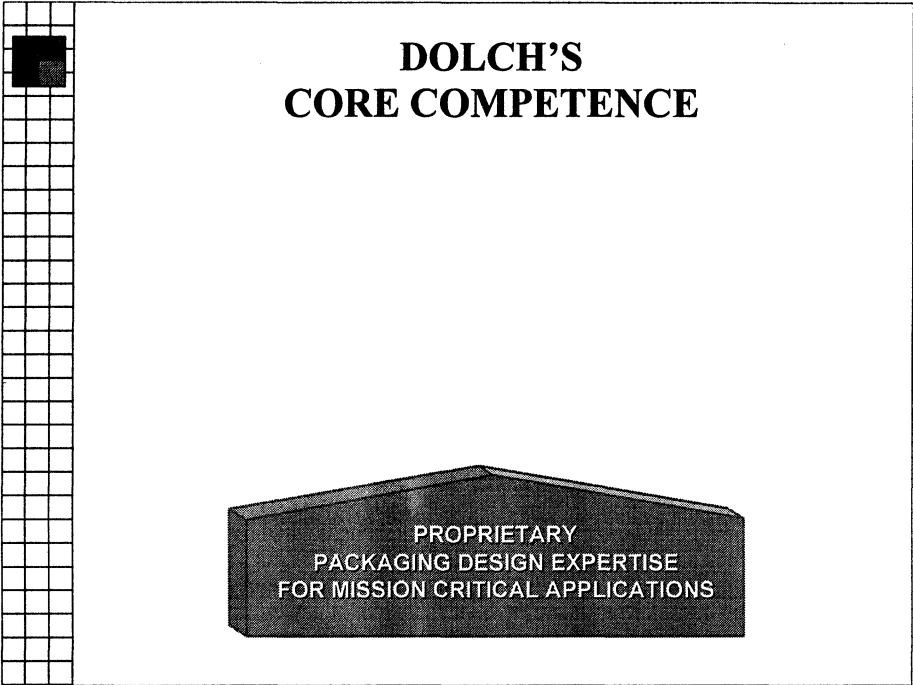
**Introduction**



Dolch  
Computer Systems

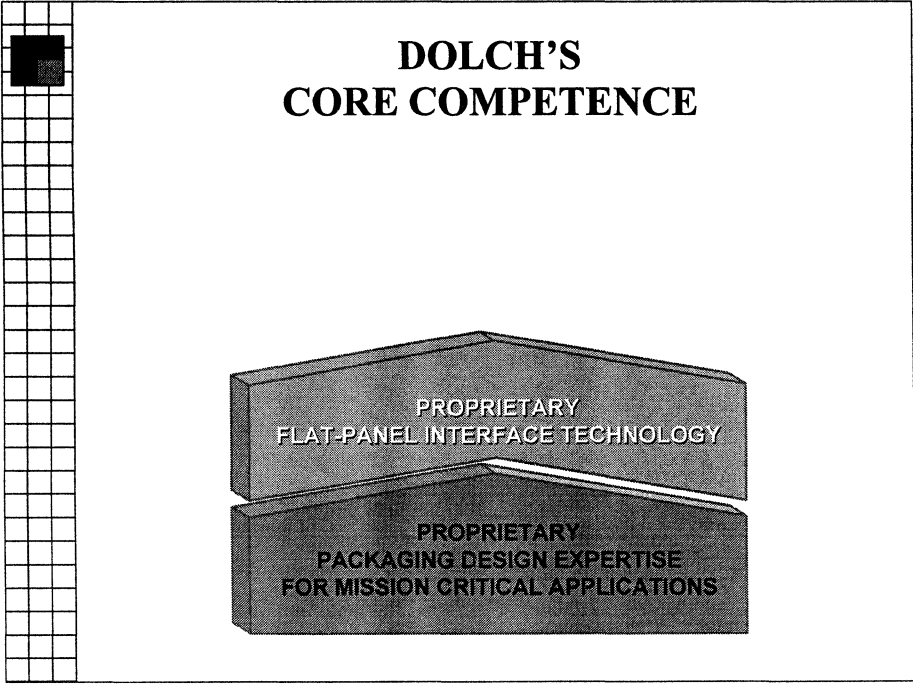
Product and Market  
Focus

# Dolch Computer Systems



**DOLCH'S  
CORE COMPETENCE**

**PROPRIETARY  
PACKAGING DESIGN EXPERTISE  
FOR MISSION CRITICAL APPLICATIONS**

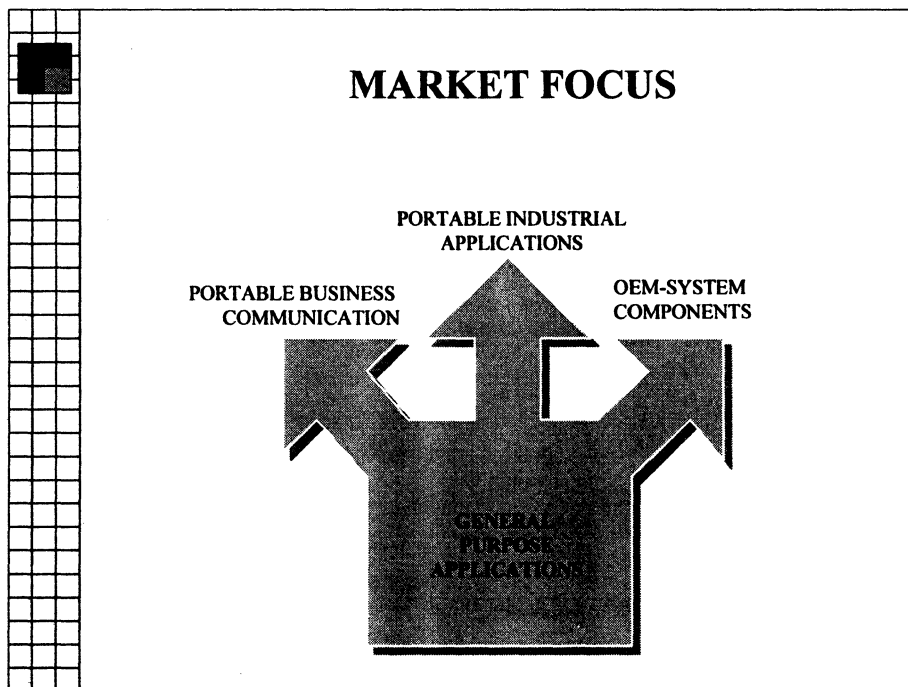
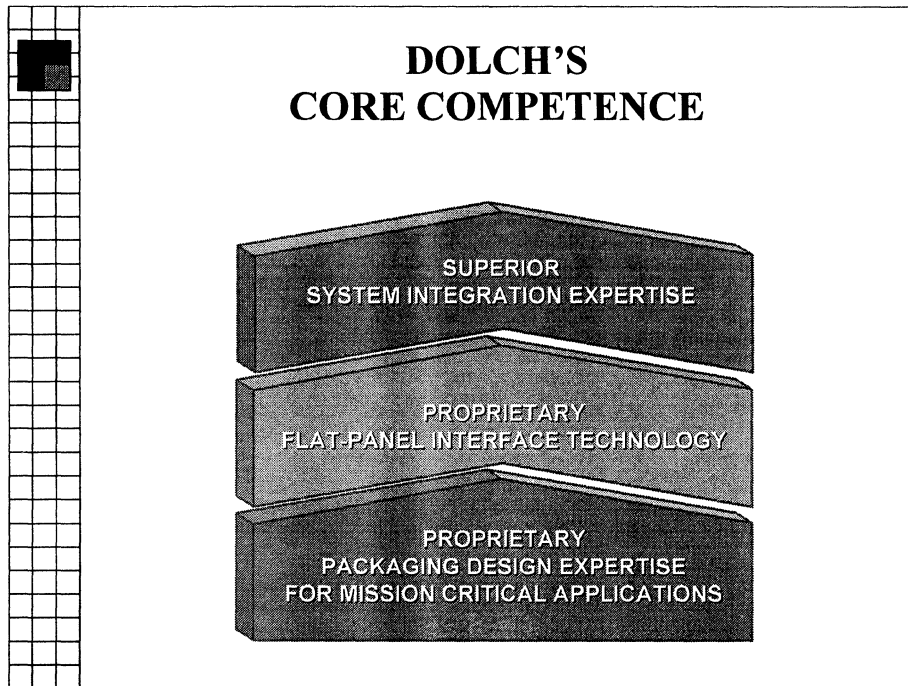


**DOLCH'S  
CORE COMPETENCE**

**PROPRIETARY  
FLAT-PANEL INTERFACE TECHNOLOGY**

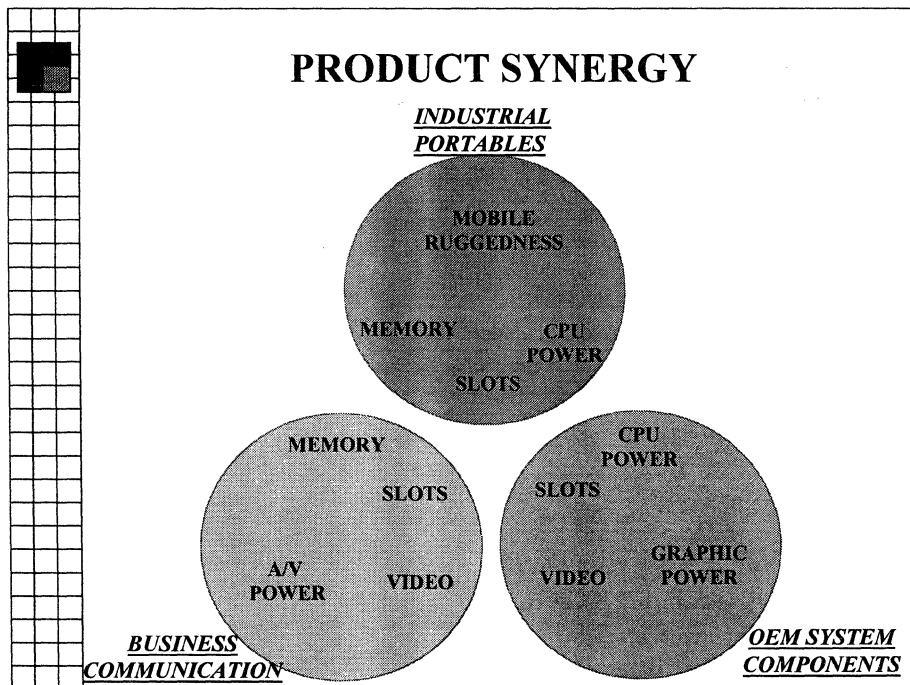
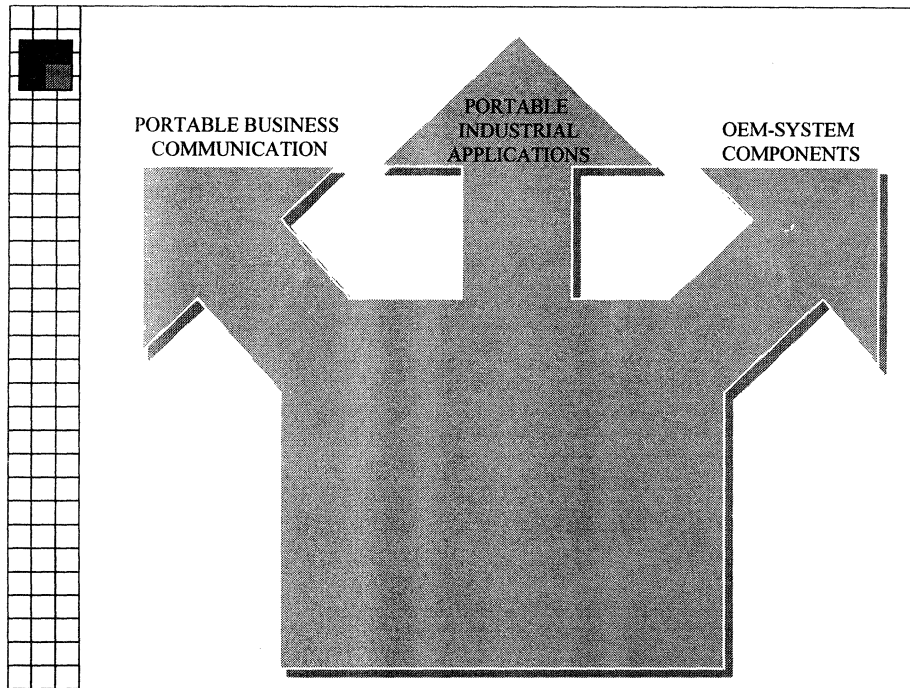
**PROPRIETARY  
PACKAGING DESIGN EXPERTISE  
FOR MISSION CRITICAL APPLICATIONS**

# Dolch Computer Systems

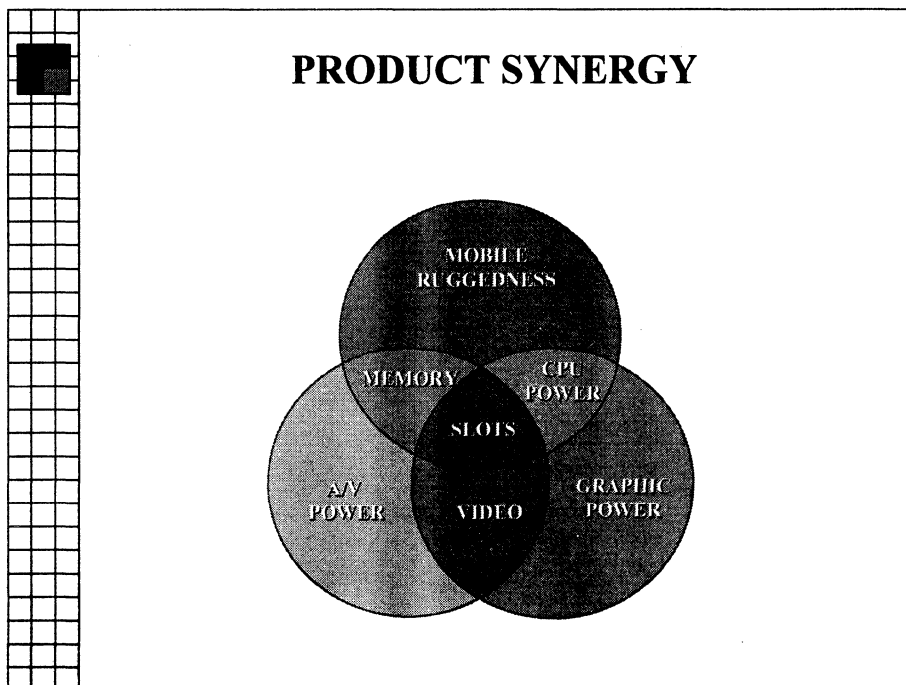
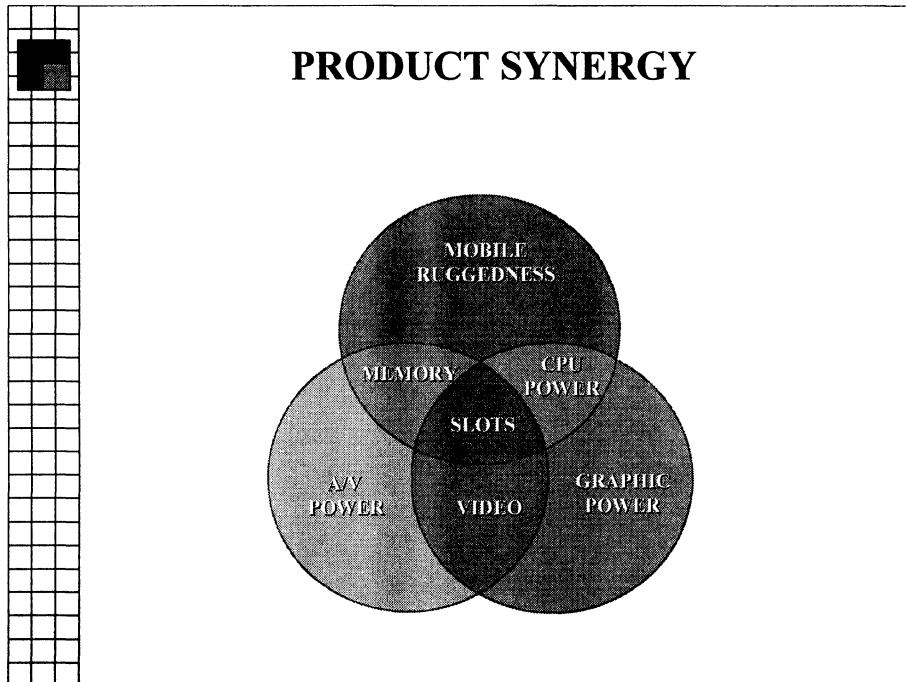




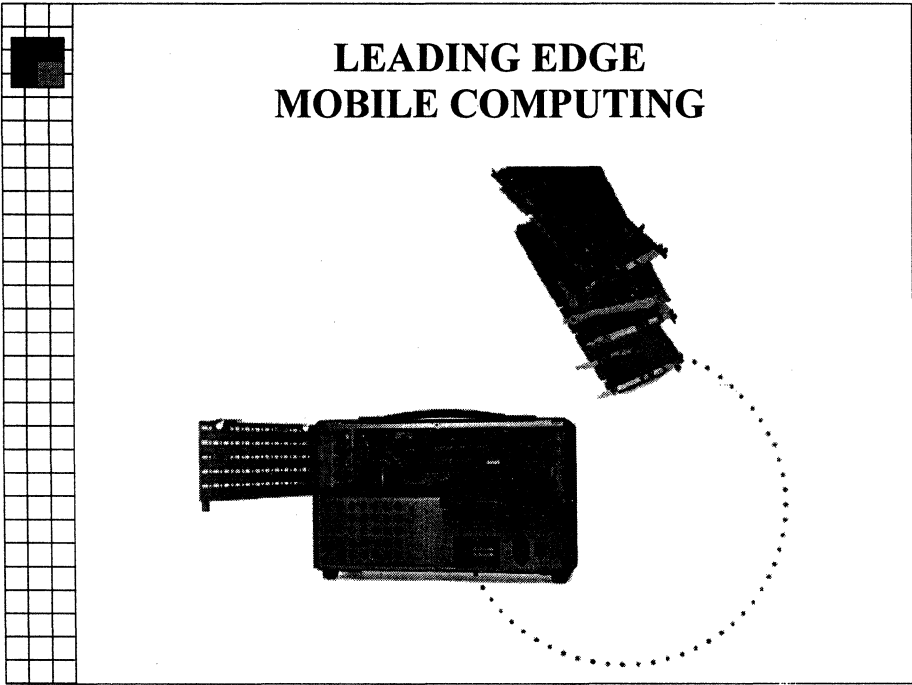
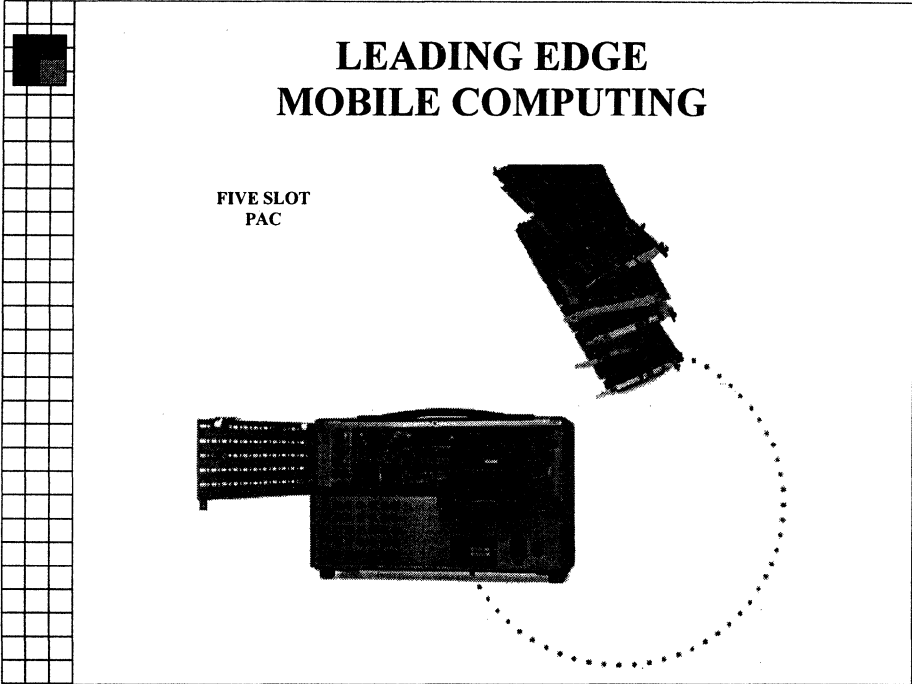
# Dolch Computer Systems



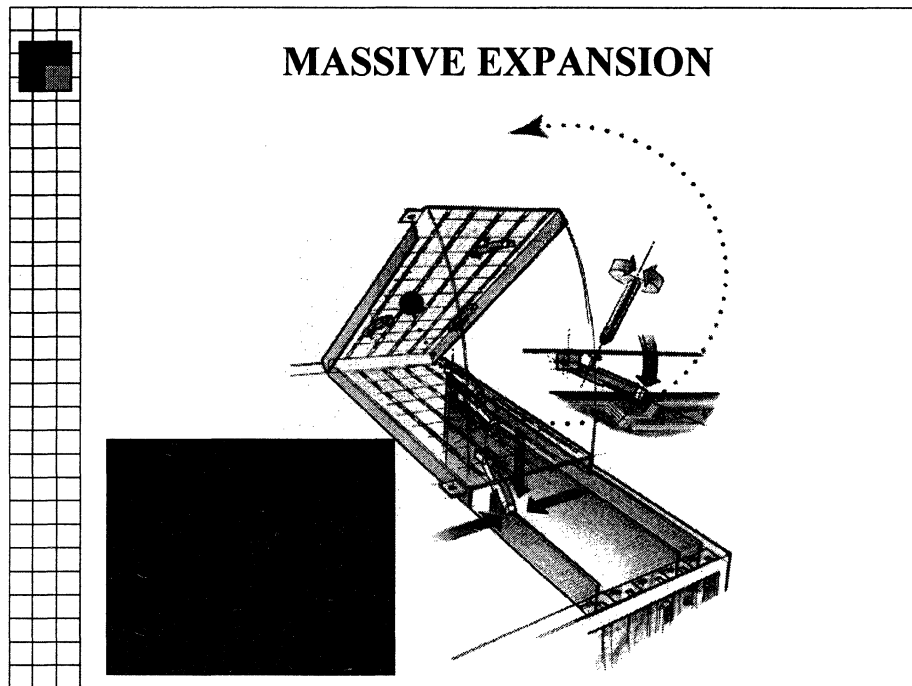
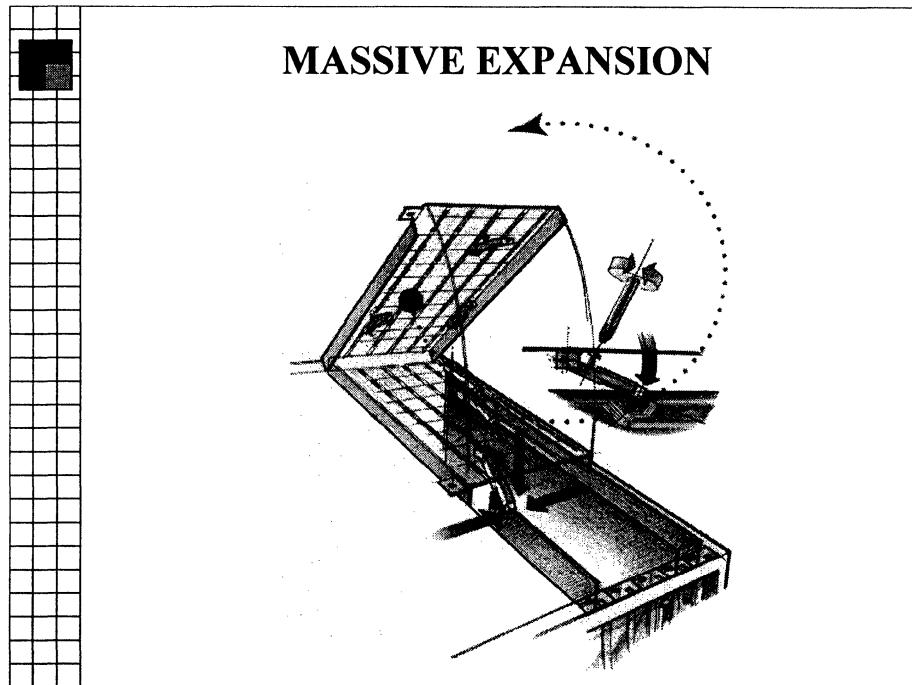
# Dolch Computer Systems



# Dolch Computer Systems



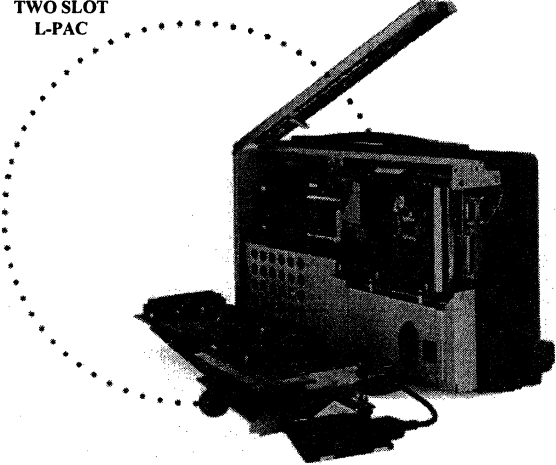
# Dolch Computer Systems




# Dolch Computer Systems

**OPTIMIZED EXPANSION**

TWO SLOT  
L-PAC




**MOBILE  
BUSINESS COMMUNICATION**



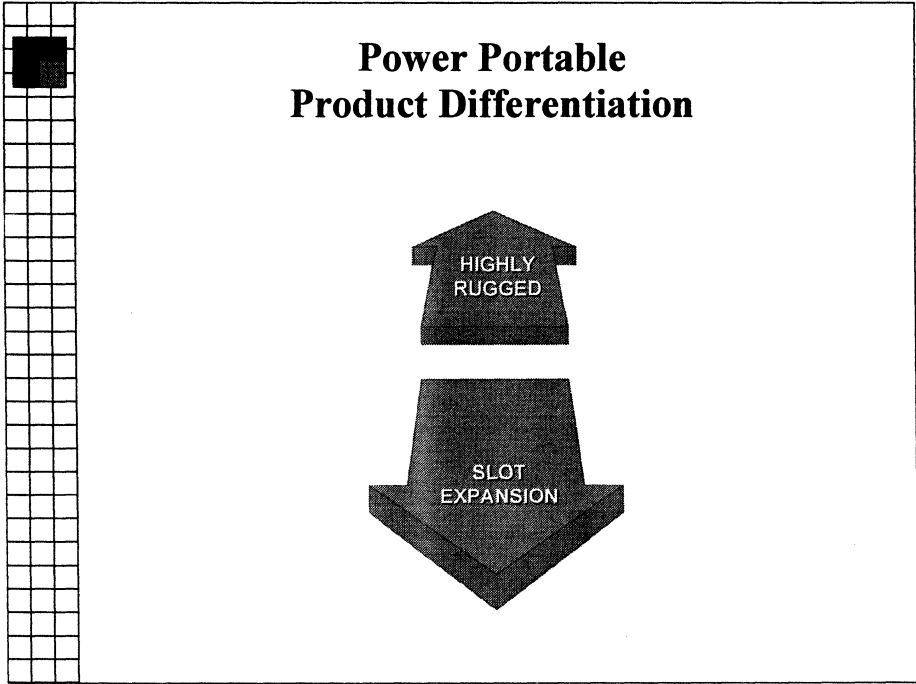
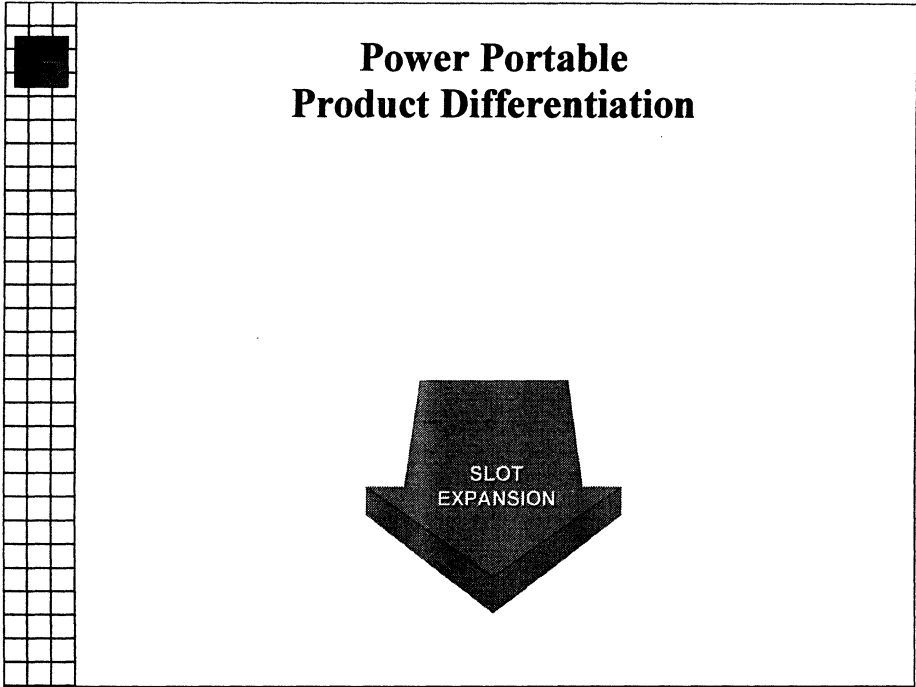
# Dolch Computer Systems

**MOBILE  
BUSINESS COMMUNICATION**

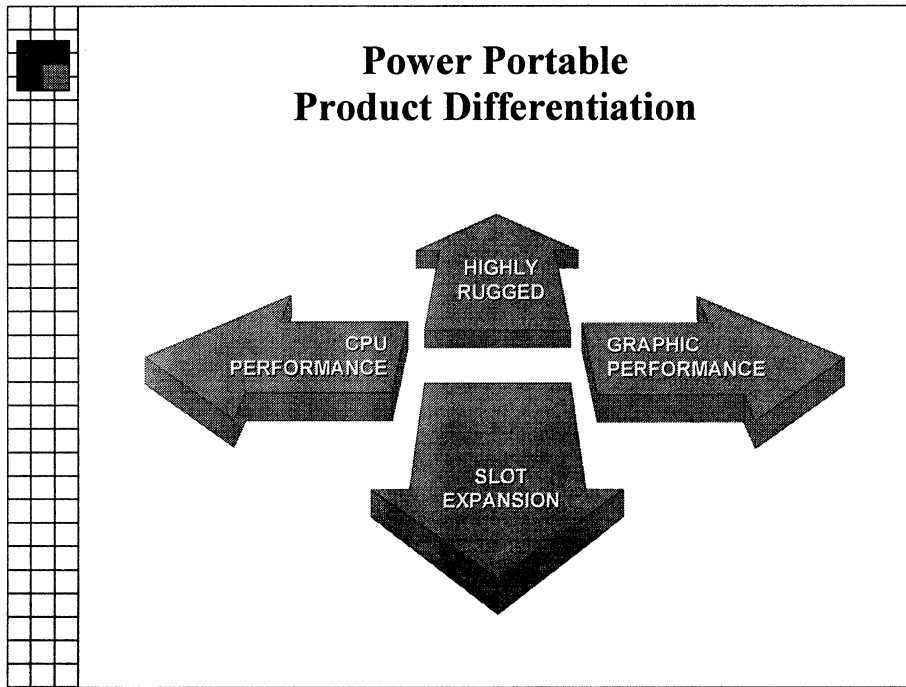
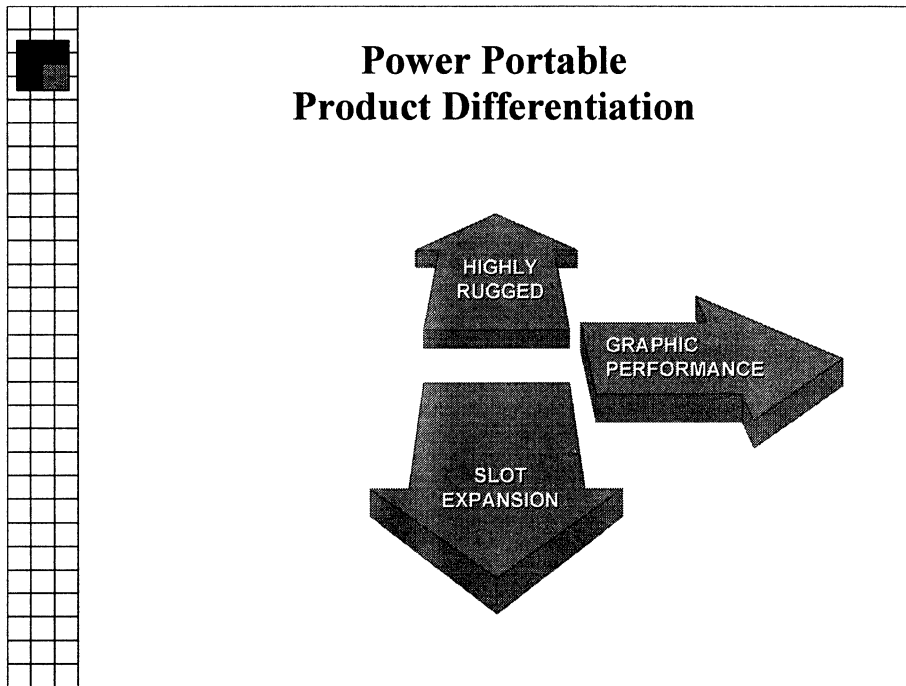


**Power Portable  
Product Differentiation**

# Dolch Computer Systems

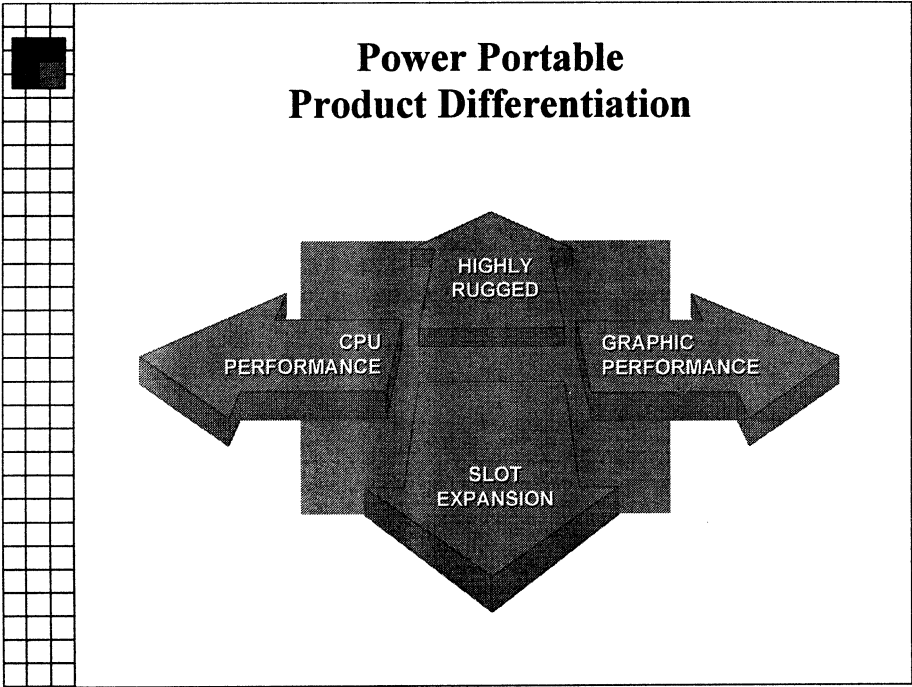


# Dolch Computer Systems





# Dolch Computer Systems



## PCI RAID CONTROLLERS

K. K. Rao  
Mylex Corporation  
34551, Ardenwood Blvd  
Fremont, CA 94555 USA  
(510) 742-7515/ (510) 797-4907 (fax)  
kkkr@mylex.com

*Abstract: This paper traces the history of Mylex RAID controller development and the migration to the PCI bus. The features of the PCI RAID controllers now in production are described along with the platforms and operating systems that are supported. Some of the issues relating to different platforms and operating systems are discussed. A look at some of the future developments is presented.*

### **FROM EISA TO PCI – A HISTORY OF MYLEX RAID CONTROLLER DEVELOPMENT**

The 90's began with expected growth in client/server solutions, and Mylex recognized the need for reliable mass storage at prices suitable for the PC server market. RAID technology offered the solution, but this necessitated the development of a host-adap-ter-based disk array controller.

Mylex RAID controller development began in June, 1991. The first product used an embedded RISC-based processor, the Intel i960CA, and was based on the EISA-bus. The controller provided five SCSI channels to interface with the disk subsystem, and it supported RAID levels 0, 1, 5, and 0+1, as well as single drive control capability ("JBOD"). Called the DAC960-5, it went into production in April, 1992.

Subsequently, 3-, 2- and 1-channel versions of this product were developed, which proved to be more cost-effective than the original 5-channel version. Further, the migration to different busses commenced.

#### **The Migration Steps to PCI**

The first migration was to the MicroChannel™ bus. The MicroChannel product, the DAC960M-2, was sampled in December, 1992 and went into production in April, 1993. The migration to a host SCSI bus, to achieve platform independence, commenced in the summer of 1993 around the same time that the development of a PCI RAID controller began.

The first Mylex PCI controller, the DAC960P, went into production in September, 1994. Subsequently, two other products, the DAC960PD and the DAC960PL, were introduced in December, 1994 and March, 1995 respectively.

To date (February, 1996), Mylex has shipped in excess of 250,000 RAID controllers, of which more than 90,000 are PCI based.

**PCI's Special Challenges.** The migration to PCI presented some special challenges, especially in terms of time to market. For both EISA and Micro-Channel busses, an off-the-shelf bus master interface controller chip was available. This made the hardware design relatively quick, since no ASIC development was needed. However, for the PCI bus, no such device was available, given that PCI started off as, (and still is, primarily), a component bus – while a RAID controller requires an entire embedded processor system.

### **RAID TERMINOLOGY**

**RAID – Redundant Array of Independent Disks**

**RAID level 0 –** Block striping is provided, which yields higher performance than with individual disk drives. There is no redundancy.

**RAID level 1 –** Drives are paired and mirrored. All data is 100% duplicated on an equivalent drive (fully redundant).

**RAID level 5 –** Data is striped across several disk drives. Parity protection provides redundancy.

**RAID level 0+1 –** Combines RAID levels 0 and 1. This level provides both striping across drives and redundancy through mirroring.

**JBOD – "Just a Bunch of Drives."** Each drive can operate independently, like with a common host bus adapter; or multiple drives may be spanned and seen as a single, very-large drive. No redundancy is provided.

**ARCHITECTURE OF MYLEX PCI RAID CONTROLLERS**

All of the Mylex PCI RAID controllers that are currently shipping share a common architecture similar to that shown in Figure 1, the block diagram of the Mylex DAC960PD.

The PCI architecture and the DRAM subsystem are the keys to the high performance of the Mylex RAID controller. The DAC960P and the DAC960PD provide two options for DRAM – a standard fast-page mode DRAM, which operates with one wait state and can provide a peak memory bandwidth of 66 Mbytes/second; and an EDRAM, which operates at zero wait states with a peak bandwidth of 133 Mbytes/ sec.

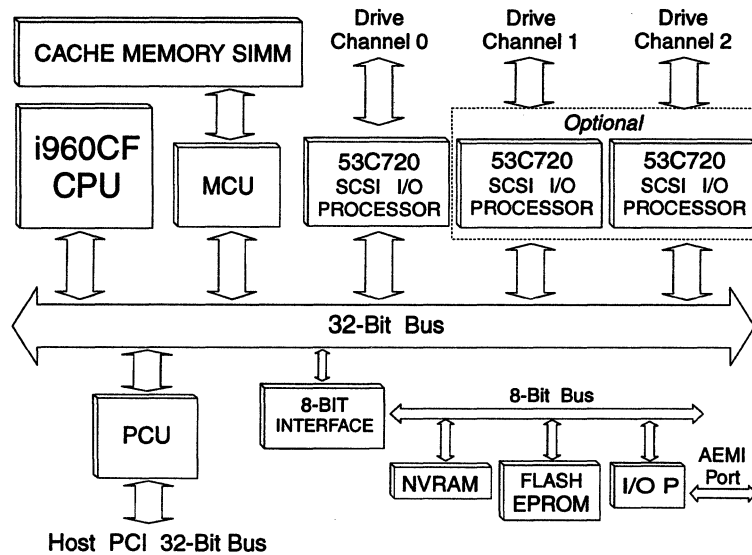
At power-up, the i960 copies code from the flash EEPROM into the DRAM and executes it from the DRAM.

Array configuration information is maintained in the EEPROM, with a backup copy in the NVRAM.

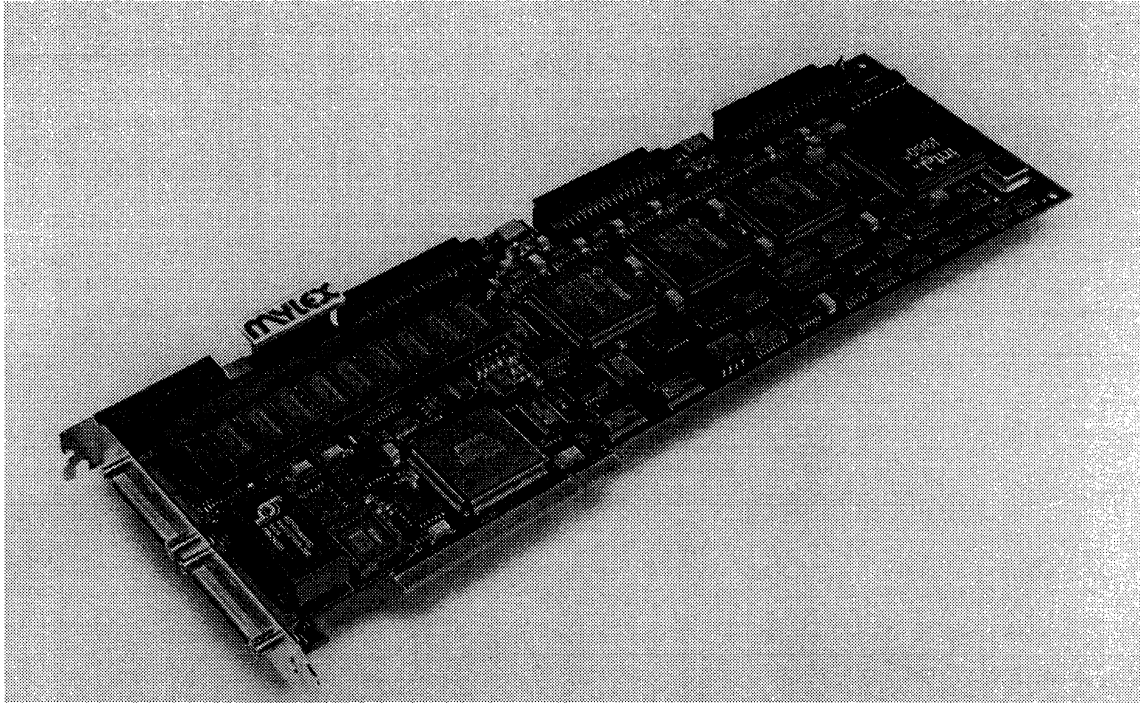
Commands from the host CPU(s) are posted into mailbox registers in the PCI interface component (the PCU). These commands are analyzed by the i960, which instructs the SCSI I/O processors (SIOPs) to initiate one or more SCSI commands to the drives.

The DRAM is also used as a cache, controlled by the memory control unit (MCU) – thus, system read commands which have cache hits result in data transfers directly from the cache to the PCI bus through the PCU. The cache can be configured as write-through or write-back on a logical drive basis.

An optional battery backup module (not shown) will maintain any write data that may be remaining in the cache in the event of a power failure.



**Figure 1. Block Diagram of Mylex DAC960PD PCI RAID Controller**



**Figure 2. Mylex DAC960PD PCI RAID Controller**

### ***PCI RAID CONTROLLER FEATURES***

The features listed below are generally common to all three Mylex PCI RAID controllers – DAC960P, the DAC960PD (shown in Figure 2) and DAC960PL. Differences between the controllers are indicated wherever applicable.

- All three models are full-size PCI boards.
- The DAC960P and the DAC960PD use an i960CF at the core. The DAC960PL uses an i960JF at the core.
- 1-, 2- and 3-channel versions are available. The basic board layout accommodates three channels. 2- and 1-channel versions are realized by depopulating the standard layout.
- All SCSI channels are Fast (10M transfers per second) and Wide (16-bit). Ultra-SCSI (20M transfers/second) versions of the DAC960P and the DAC960PD are currently sampling, and will be in production in April, 1996.
- Cache memory options are 2, 4, 8, 16 and 32 Mbytes with fast-page mode DRAMs, and (not available for the DAC960PL) 4 and 8 Mbytes with EDRAMs.
- Data transfers between cache memory and the PCI bus are sustained at almost 133 Mbytes/second with EDRAM and almost 66 Mbytes/second with DRAM.
- The battery backup module option is available for all models.
- The DAC960P has a single 68-pin high-density connector on the back-plane for connecting one SCSI channel externally. The DAC960PD and the DAC960PL have two 68-pin ultra-high-density connectors on the back-plane providing for two external SCSI channels.
- Supported RAID levels are 0, 1, 5, 0+1 and JBOD.
- Multiple disks can be designated as hot spares, to dynamically replace failed drives.
- Hot swapping of disk drives is supported.
- A pass-through mode is available for non-disk devices, such as CDRoms and tape drives.
- Multiple-drive subsystem enclosure management schemes are supported: Digital StorageWorks™, Conner, Mylex AEMI, SAF-TE, as well as some proprietary schemes for key OEM customers.
- The DAC960P and the DAC960PD can deliver 29 Mbytes/second of sustained sequential read performance from the disk drives to the system memory, and 2000 iops random performance.

**OPERATING SYSTEM AND PLATFORM SUPPORT**

In general, support for an operating system consists of a RAID controller device driver and one or more administration and monitoring utilities which notify system administrators of hardware, software, or network-related events requiring attention.

Mylex development efforts are on-going in support of the major client-server operating environments and platforms.

**Intel (x86) Architecture Platforms**

The Intel x86-based operating systems that are supported are:

- Novell Netware 3.1x and 4.x,
- Microsoft Windows NT 3.51,
- IBM OS/2 2.x and 3.x,
- SCO ODT 3.0 and Openserver Rel 5.0,
- Unixware 2.01,
- Banyan Vines 6.0.

Administration and monitoring utilities for all the above are server-based. In addition, a client/ server-based GUI utility, the Global Array Manager™, is available for the following servers and clients:

**Servers**

- Windows NT 3.51
- Novell NetWare 3.1x and 4.x

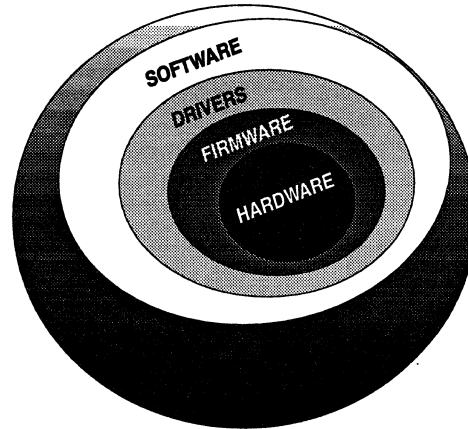
**Clients**

- Microsoft Windows
- Windows for Workgroups
- Windows NT.

Initial array configuration is through a DOS-based utility (DACCF.EXE). Booting from the array is possible through the DAC960 BIOS.

**PowerPC Platforms**

The supported operating system for the PowerPC is Microsoft Windows NT 3.51. Initial array configuration is through an Open Firmware based utility. Booting from the array is through an Open Firmware boot ROM.



**Figure 3. Mylex RAID Controller Architecture**

**Digital Alpha™ Platforms**

Supported operating systems for Digital Alpha platforms are:

- Microsoft Windows NT 3.51,
- OSF-1,
- VMS.

Initial array configuration is through an ARC (Advanced RISC Computing standard) based utility. Booting from the array requires support from the system firmware and is available on several platforms.

**MIPs R4000 Platforms**

The supported operating system for MIPs R4000 platforms is Microsoft Windows NT 3.51. Initial array configuration is through an ARC based utility. Booting from the array requires support from the system firmware and is available on several platforms.

**PCI-RELATED ISSUES WITH PLATFORMS AND OPERATING SYSTEMS**

Most of the PCI-related issues that Mylex faced when developing for different platforms and operating systems were due to the relative infancy of PCI. Some issues were due to the ambiguity in the 2.0 PCI Specification. The 2.1 Specification addressed many of these issues.

**Protocol and Timing Issues**

The initial PCI implementations in the Intel architecture – the Mercury and Neptune chipsets – were aimed primarily at the desktop markets. System memories could only support transfer rates in the order of 40 to 50 Mbytes/second. Thus, the full PCI bandwidth was not being tested at that time. With the emergence of the newer Triton chipsets, that was no longer the case, giving rise to some protocol and timing problems.

The 2.0 Specification indicated that accesses to registers on devices had to be provided in both memory and I/O spaces in order to support processors which did not have explicit I/O instructions. However, this was mentioned at only one place in the specification and was overlooked by many implementers causing problems in porting to non-x86 platforms. The 2.1 Specification came out with a very clear recommendation to implementers to provide both kinds of accesses to registers.

In installations with multiple adapters from the same vendor, the addressing of specific controllers from a utility and user interface perspective is an issue. The 2.1 BIOS Specification provides a new call which returns a slot number given a bus, device and function number. However, in operating systems such as Windows NT, where BIOS calls are not possible and PCI configuration information is provided through operating system functions, the physical slot number cannot be obtained.

**Boot Device Order Issues**

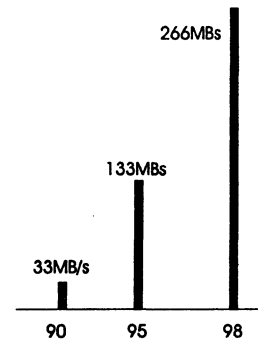
Boot device ordering, when using heterogeneous host bus adapters, is another major issue. In earlier busses (e.g. ISA, EISA and MicroChannel), the boot device could be selected by the user through configuration options.

One of the advantages of PCI is that it is intended to function as plug-and-play, thus eliminating the need for users to deal with configurations, BIOS addresses, and so on. A disadvantage of automatic configuration is that user flexibility is lost. For example, it is entirely possible that the boot device can change when a user inserts a new adapter in a system.

Typically, there is no way for the user to change the boot device in a PCI system. Mylex has addressed this issue by providing an INT13 support enable/disable feature.

**FUTURE DEVELOPMENTS — FUTURE CHALLENGES**

The current PCI bandwidth of 133 Mbytes/sec is fairly adequate for the state-of-the-art in storage technology (Ultra SCSI, 40 Mbytes/ second), even for a 3-channel RAID controller product. The challenges in achieving *maximum bandwidth* on the storage bus through a RAID controller are bigger than those related to *saturation* the PCI bus.



**Figure 4. RAID Controller Bandwidth Needs**

As newer and faster interfaces such as Fibre Channel, SSA and Fast-40 SCSI begin to appear, however, the PCI bus will bottleneck easily at its present 32-bit, 33 MHz rate. This will necessitate a migration to a higher bandwidth bus, either through a faster clock, 66 MHz, or increased width, 64-bit.

The former appears to be a lower-cost alternative, since there will be no need to increase pin count on IC packages. However, the challenges in making a system work at 66 MHz with adapters from different vendors can be significant. Further, the lack of backward compatibility can be another issue – a 66 MHz PCI bus will run at 66 MHz only when all devices and adapters on it are 66 MHz capable.

The advantage of a 64-bit PCI bus is that it is backward compatible – devices which are 64-bit capable (typically, system memory and the bus master in question) can communicate at the faster rate, allowing all other devices to operate at their own rate. On the other hand, the disadvantages of 64-bit PCI are the costs associated with increased pin count and the board layout issues when dealing with the wider bus.

**CONCLUSION**

Future needs for larger bandwidth for PCI RAID controllers can be satisfied relatively easily by means of the 64-bit PCI bus. This conclusion is based on experiences that Mylex has encountered during the development of RAID controllers for different host busses since 1991; and validated during and the production of over 250,000 controllers to date (specifically the three PCI RAID controllers that Mylex has been shipping for some time), to support a wide range of operating systems and platforms. Up to now, most of the issues with PCI were due to the relative infancy of the PCI specification and have been addressed in the version 2.1 specification.

**ABOUT THE AUTHOR**

**K. K. Rao**, Vice President of Engineering, Mylex Corporation, Fremont CA. Mr. Rao has been leading the Mylex RAID program since joining the company in 1991. Prior to joining Mylex, Mr. Rao was responsible for hardware and firmware design at Processor Systems, India. He holds MSEE and BSEE degrees from the Indian Institute of Technology, Bombay.

PCI Conference/May 1996  
Embedded RAID presentation

**Abstract:** RAID has gained in popularity and acceptance in the last two years. Many recent improvements in features and architecture have increased the performance and dramatically affected the price/performance for both entry level and midrange server RAID options. This presentation examines the current market for creating a scaleable embedded I/O architecture for RAID that offers low cost, high performance and compliance with current industry standards.

**Outline**

**Market Issues**

- I/O architecture
- Industry Standard Architectures (PCI)
- Cost/Performance

**Why RAID on the Motherboard**

- Demand for architecture
- Dealing with cache issues

**Technology Issues**

- Silicon development
- Software and engineering
- The interface
- Utilities and the GUI
- Remote management

**The Future**

- RAID on the desktop
- Scaling the architecture
- Serial I/O



## **FAST-40 SCSI, PUSHING PCI TO THE LIMIT**

Richard Mourn  
Symbios Logic Inc.  
1635 Aeroplaza Dr.  
Colorado Springs, CO 80916  
richard.mourn@Symbios.com

### ***Introduction***

Since the advent of the personal computer the increase in bandwidth of both the main processor and I/O devices is substantial. On the processor side, raw bandwidth has been increased from 10 MB/s to 1064 MB/s in a little more than 10 years. In early systems floppy drives were the fastest devices attached. Soon after I/O devices like IDE and SCSI hard disk drives started pushing the limits of the ISA bus with burst speeds of less than 10 MB/s. As a result of this, migration of the host bus has happened as well; from the XT to AT/ISA to EISA to Microchannel to PCI. The increase in bandwidth from 4 MB/s to 132 MB/s is substantial. However, PCI is unique in its acceptance. It has been adopted in low end PCs to the highest end workstation/servers and even in subsystems such as RAID and telecom boxes. The need to perform, yet be cost effective, is evident. Today 32 bit 33 MHz PCI has more than enough bandwidth for lower end PCs, but higher end systems are starting to push the limits once again. This push is being intensified by higher and higher performing video requirements and is also very pronounced in the I/O arena where both ANSI and the IEEE have developed new I/O interface standards that have surpassed 10 MB/s and are starting to exceed data rates of 100 MB/s. These I/O interface standards include Ultra-2 SCSI (Fast-40), Fibre Channel, IEEE-1394 and SSA, which operate in the 40 to 100 MB/s range. In this paper Ultra-2 SCSI, the newest of the four mentioned above, will be the topic of discussion. However, most of the system level issues touched on in this paper apply to any I/O device, and will become more important as these new I/O technologies approach 60% or more of the 32 bit 33 MHz PCI buses theoretical bandwidth of 132 MB/s.

### ***Ultra-2 SCSI (Fast-40) Overview***

Ultra-2 SCSI is an incremental step in the migration of SCSI to higher and higher data rates. ANSI X3T10 SPI-2 Study Group is defining this addition to the SCSI-3 standard. The primary difference between Ultra-2 SCSI and its predecessors is the adoption of Low Voltage Differential (LVD) transceivers which allow synchronous data transfer of greater than 80 mega-transfers per second at cable lengths of 12 meters with 15 LVD devices attached. The transceivers can be implemented using generic CMOS processes which allow the integration of the transceivers and the SCSI processor. This enhancement provides the connectivity, distance and reliability of RS-485 based differential SCSI without the space and cost penalty.

The software changes required to implement Ultra-2 SCSI are very similar to the migration from Fast SCSI to Ultra SCSI (Fast-20), mainly synchronous negotiation. However, the key is to create a device that has the bandwidth to fully support Ultra-2 SCSI's 80 MB/s and not over burden the host bus (PCI). An intelligent implementation is a must. Intelligence here assumes very little assistance from the host to complete an I/O.

## *Bandwidth*

From a system level point of view the performance increase when migrating from Ultra SCSI, which when running wide transfers has a peak synchronous transfer rate of 40 MB/s, to Ultra-2 SCSI, which when running wide transfers has a peak synchronous transfer rate of 80 MB/s, is considerable. With 60% of the potential PCI bandwidth being consumed by one device, the chip hardware, software, and PCI chip set designers must pay particular attention to each segment of their design to insure optimum performance. In early tests, Symbios Logic has found that PCI chip sets limit wide Ultra-2 SCSI performance. When running Ultra-2 SCSI narrow the SCSI bus is limited to 40 MB/s synchronous. In the test environment 37 MB/s was achieved using a low latency target. Running this same test, only this time using wide transfers, resulted in 56 MB/s out of a possible 80 MB/s available on Ultra-2 SCSI bus. A few of the questions posed by this paper are: How far can the current 32 bit, 33 MHz architecture be perfected before most systems require 64 bit or 66 MHz PCI or both? What can host implementations do to prolong its life?

## *PCI Arbitration*

Before the meat of an I/O transfer can be looked at, how the PCI bus handles arbitration is important. On the PCI bus, arbitration is controlled by a central entity known as the bus arbiter. The arbiter is usually physically located in the PCI chip set. Specifically, it may be integrated into the host/PCI or the PCI expansion bus bridge device. The PCI specification does not define the algorithm used by the bus arbiter to decide the winner of a arbitration. The bus arbiter can use any scheme as long as it is fair and avoids deadlock. This vagueness allows many different implementations and provides an area for improvement. The arbiter could implement a fairness algorithm based on fixed order, or rotationally or a combination of them both.

To describe PCI bus arbitration in short, when a PCI master wishes to gain control of the PCI bus it asserts its REQ# signal. The bus arbiter sees the request and based on the algorithm used to determine ownership grants the bus to the requesting master by asserting GNT#. Arbitration Latency is the time from the assertion of REQ# to the assertion of the corresponding GNT#. This time period is highly variable and can take several clocks. In cases where only one device is competing for the bus, 10 PCI clocks have been observed. When the device receives the grant it cannot begin its cycle until the PCI bus goes idle. This time is known as Bus Acquisition Latency and is defined as the time from the reception of GNT# to when the current master surrenders the bus. This is controlled by the current master's Latency Timer expiration or completion of its data transfer. The last piece of this puzzle is the Target Latency which is the time from the beginning of the transfer cycle to the beginning of the data transfer. The total Bus Access Latency is the summation of the Arbitration Latency, Bus Acquisition Latency, and Target Latency.

The PCI specification has placed several hooks that allow Bus Access Latency to be controlled and those hooks are being put to better use in newer designs. The use of the maximum latency, minimum grant, and Latency timer are describe here and are very useful if the bus arbiter is programmable. The programmable bus arbiter should be configured by configuration software at startup. The configuration software determines the priority to be assigned to master capable members of the PCI bus. This is accomplished by reading each members maximum latency (*Max\_Lat*) and minimum grant (*Min\_Gnt*) registers. Each bus master (the Ultra-2 SCSI device) indicates in these registers how quickly it requires access to the bus and how long it would like to retain the bus in order to achieve its required performance. The smart programmable bus arbiter based on this information, programs the Latency Timer in the device and establishes the priority each device will have in its arbitration algorithm. Therefore it is imperative that master capable devices set the *Max\_Lat* and *Min\_Gnt* registers to values appropriate for their device.

## I/O Transfers

In this example a single SCSI I/O is broken down to better understand how an intelligent Ultra-2 SCSI I/O device can push current PCI bus implementations to the point where it hinders the Ultra-2 SCSI bus performance. It is important to point out here that non-intelligent implementations will over burden the system and will not allow 80 MB/s over the SCSI bus and will reduce the performance of other devices connect to the PCI.

The system used for this experiment consisted of a 100 MHz Pentium machine with 32 MB of memory. The memory sub-system was able to receive 64 dword bursts without inserting wait states. The PCI bus was loaded with three devices (host bridge, video, and SCSI). The SCSI bus controller was mostly intelligent with all hardware specific code running local and all data structure information for the I/O was running in host memory. The target device emulates a heavily loaded SCSI bus which is sustaining as close to 80 MB/s as possible. It will be paced by the initiator. A 1 mega-byte file is being transferred from the target to the initiator. This file is broken down into 32 - 32 KB SCSI transactions. The initial seek time is 5 ms but the remainder of the transaction is completed with a minimal latency between disconnects.

The purpose of this example is to show how an Ultra-2 SCSI device can and does exceed the effective data transfer rates of the PCI system under test. This example also illustrates how an intelligent Ultra-2 SCSI device reduces the PCI traffic for non-data transfers and increases the data throughput of the Ultra-2 SCSI device. The goal is to have 80 MB/s across SCSI and PCI.

Table 1 describes the flow of a simple SCSI read transaction and will provide the baseline for understanding the bandwidth required to meet Ultra-2 SCSI's needs.

SCSI Phase	Description	Time
Arbitration	Time from BUS FREE through the time when the Initiator has won the SCSI Bus.	~5µs
Selection	Time from Arbitration through the time when a device has been selected and asserted BUSY	~0.6µs
Message Out	Initiator sends Identify message (assume synch done)	~0.1µs
Command	Initiator sends Read Command to target RAID	~0.5µs
Message In	Target Disconnects from Initiator	~0.2µs
<b>Sub Total</b>		<b>6.4µs</b>
<b>Disconnect Time</b>		<b>5ms</b>
Arbitration	Time from BUS FREE through the time when the Target has won the SCSI Bus.	~5µs
Reselection	Time from Arbitration through the time when the initiator has been selected and asserts BUSY	~0.6µs
Message In	Target sends Identify message	~0.1µs
Data In	RAID sends data to Initiator (32 KB transfer)	~400µs
Message In	Save Data Pointers and Disconnect	~0.2µs
<b>Sub Total</b>		<b>405.9µs</b>
*****	<b>Repeated 31 Times</b>	*****
Disconnect Time		1µs
Arbitration	Time from BUS FREE through the time when the Target has won the SCSI Bus.	~5µs
Reselection	Time from Arbitration through the time when the initiator has been selected and asserted BUSY	~0.6µs
Message In	Target sends Identify message and Restore	~0.1µs

	Pointers	
Data In	RAID sends data to Initiator (32 KB transfer)	~400µs
<b>Sub Total</b>		<b>12.6ms</b>
*****	<b>End Repeated 15 Times</b>	*****
Status	Target sends status to initiator	~0.1µs
Message In	Target sends complete message	~0.1µs
<b>Total</b>		<b>13 ms + 5ms</b>

**Table 1. Example SCSI Transfer.**

This SCSI I/O takes 13 ms plus the 5 ms seek latency to transfer the 1MB file (76.92 MB/s). This is very consistent with the actual data obtained using the Ultra-2 SCSI device doing narrow transfers. If you remember 37 MB/s was achieved from a possible 40 MB/s. The SCSI overhead for both narrow and wide transfers will remain constant. The following paragraphs will examine some of the system overhead that is needed to generate and service this request, and how that translates into PCI bus latency.

### **Host Based Operation**

The application makes a read request to the operating system (OS). The operating system sends the request to the SCSI device driver which translates the I/O request into a data structure. This structure consists of a scatter/gather list, pointers to the SCSI message and command, and where the SCSI status will be written. Even though the SCSI bus transaction has not been started, **hundreds** of PCI transactions have already taken place. Reads from the hard disk drive's FAT and sector tables, host buffer allocation, and memory transaction have been executed to allow the SCSI driver to formulate this data structure. The PCI chipset, by supporting Memory Write & Invalidate and Read Multiple ensures that these transactions are done optimally. The I/O device driver needs to ensure that the translation from the OS's I/O request to the data structure is as clean as possible.

### **SCSI Controller Operation**

At this point, the implementation used to execute this SCSI I/O is crucial to maintaining PCI bandwidth optimization. Good SCSI devices are "intelligent" and do not require constant attention. An example of an intelligent device would be one that requires less than one interrupt per I/O and only needs to access the PCI bus for start up, data transfer, end of I/O, and perhaps some exception handling. Non-intelligent devices would require assistance in servicing two or more of the SCSI phases and must go across the PCI bus to fetch all instructions and data structure information.

Symbios Logic's SYM53C8xx family of SCSI controllers allow for designs that can operate with one interrupt per I/O and only need to access the PCI bus for startup, data transfers, I/O completion, and some exception cases. This sort of performance advantage requires the following features:

- 1) On chip or on board intelligence (RISC processor).
- 2) On chip or on board memory dedicated to processor code and context information.

Without intelligent Ultra-2 SCSI devices for host applications the host processor, memory, and the PCI bus can be over burdened by just one I/O connect.

## **Intelligent Implementation**

Once the data structure information is formulated it is loaded from host memory. The SCSI device requires one more write to begin the SCSI bus I/O transaction. When the starting address has been written to the SCSI device, the first instruction can be fetched from local memory. In this case the first command would be a Select with Attention. This starts the SCSI bus arbitration, see Table 1. Once the device has won arbitration and selection has taken place, each SCSI command is fetched from local memory. When the disconnect message is received, the SCSI processor saves the relative information and it proceeds to another I/O or waits to be reselected. Once reselection takes place the target identifies itself, which enables the SCSI processor to reestablish the nexus for this I/O. As data is received the scatter/gather buffer information is retrieved from host memory and the data is written to host memory across the PCI bus. Again the target disconnects from the SCSI bus and the relative information is saved to host memory. For this example this process repeats itself 31 more times and on the last transfer the target finishes normally with status and the target responds with the completion message. The initiator then interrupts the host telling it that the I/O is complete.

In this intelligent implementation example the PCI bus was only used for data structure updates, data transfer and only one interrupt had to be serviced. The number of PCI clock cycles required to service an interrupt is OS dependent but typically takes about 2500 PCI clock cycles to service. If the average PCI bus access latency is  $2.5\mu\text{s}$  and data is burst at a rate of 64 dword per ownership it would consume another  $1.94\mu\text{s}$ . The time required to transfer the entire 1 MB file would be 18.19 ms (54.97 MB/s). This again is very consistent with the actual data taken.

## **Conclusion**

Many of the PCI systems out there today will not support sustained transfer rates of 75 MB/s and above. However, it not as bleak as it might seem, most I/Os are bursty and do not require very high sustained data rates. For applications where high sustained rates are needed, two PCI buses will most likely be implemented; one for slow devices (32 bit, 33 MHz) and one for higher bandwidth applications (64 bit, 33 MHz or 32 bit, 66 MHz ). From an I/O implementor's point of view the following design feature should be supported:

- 1) Intelligent implementation:
  - Reduce the number of interrupts per I/O to 1 or less.
  - Optimize such that the PCI bus is used for data transfer, not fetches of commands and data structures.
- 2) Use the PCI performance commands
  - Memory Read Line or Memory Read Multiple
  - Memory Write and Invalidate
- 3) Support bursts of at least 256 bytes
- 4) Appropriately set the Max\_Lat and Min\_Gnt registers and allow the Latency Timer to be set accordingly.

If all devices implement these features the PCI bus will be optimized for data transfer, latency will be predictable, and I/O devices will work at their performance requirement, not PCI's limitations.

**Abstract:**

This white paper discusses the system level issues encountered when connecting high performance I/O devices, such as Ultra-2 SCSI (Fast-40), to PCI. With the theoretical maximum bandwidth of a 32 bit, 33 MHz PCI bus being 132 MB/s one might not expect that most systems cannot support the 70-80 MB/s sustained data rate of a Ultra-2 SCSI interconnect. This paper walks thru a SCSI I/O and looks at the areas where intelligent SCSI implementations reduce overhead traffic on PCI and increases the data through put.

**What's Good  
&  
What's Bad  
About Unified Memory Architectures  
(UMA)**

268

Desi Rhoden  
Manager, Strategic Programs  
VLSI Technology, Inc.  
8375 S. River Parkway  
Tempe, AZ 85234  
602-752-6323  
desi.rhoden@vlsi.com

## Unified Memory Architecture

### ***What UMA Should Be***

---

- ◆ **Transparent to all OS's and applications**
- ◆ **Equal or better performance than non-UMA**
- ◆ **Compelling value to the industry**
- ◆ **Simple to implement and use**



## Unified Memory Architecture

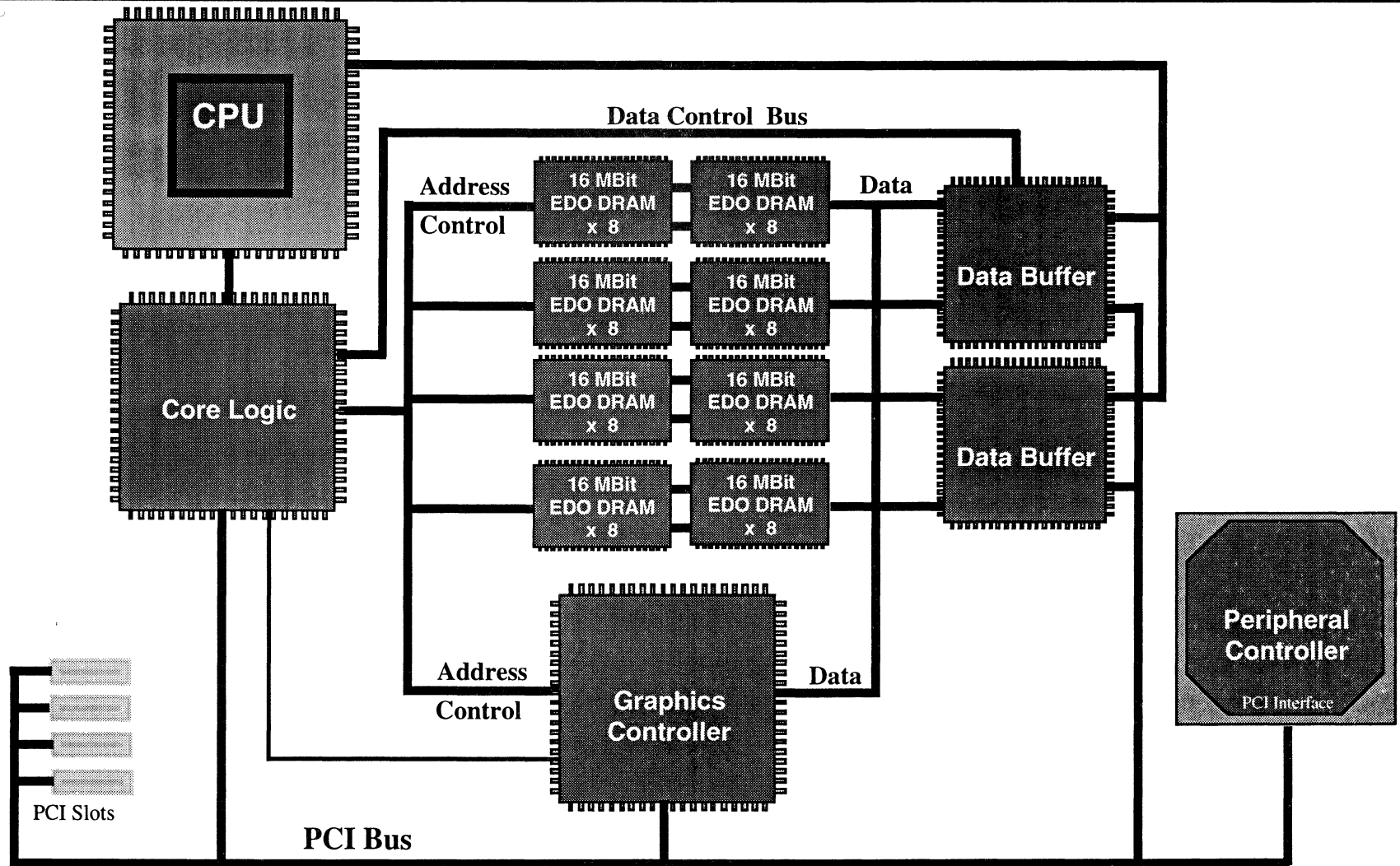
### ***What's Bad about UMA***

---

- ◆ **Asynchronous memory technologies**
- ◆ **Sharing memory control signals at high frequencies**
- ◆ **Exposing low level hardware to applications**
- ◆ **Multi-chip solutions do not off load PCI**
- ◆ **@ 16MBit memory, UMA is not compelling**

# Unified Memory Architecture

## *Current Industry State of UMA*



271

## Unified Memory Architecture

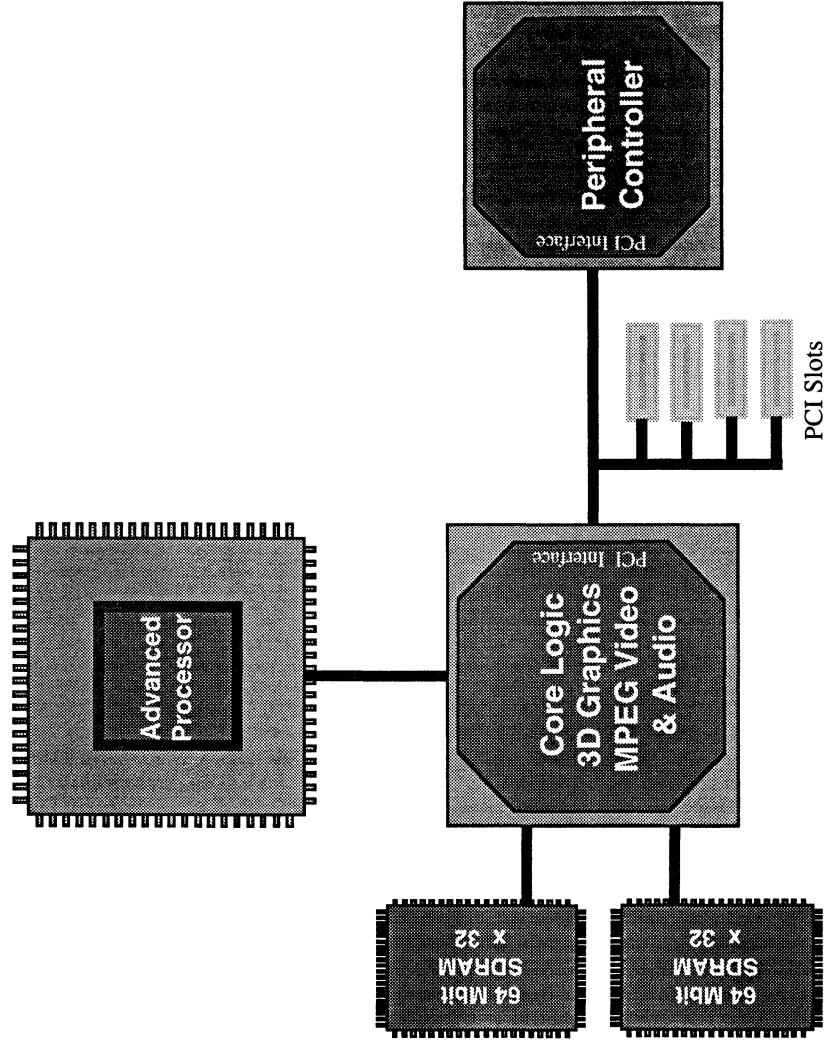
### ***What's Good About UMA***

---

- ◆ **Addresses the increase in memory granularity**
- ◆ **Can put graphics (et.al.) at the core of the machine**
- ◆ **Can be transparent to all OS's and applications**
- ◆ **Integrated solutions can off load the PCI bus**
- ◆ **Enables even higher system integration**
- ◆ **@ 64MBit memory, UMA is compelling**

272

# Unified Memory Architecture *Fully Integrated UMA*



**“Leveraging PCI Bus Bandwidth and High Performance CPUs in Designing  
MPEG-1 and H.261 Video CODECs”**

Frank Schapfel  
Multimedia Engineering Manager  
Digital Equipment Corporation  
77 Reed Rd. HL02-1/H12  
Hudson, MA 01749  
(508) 568-4861/6371 (fax)

Digital Semiconductor, a Digital Equipment Corporation business, headquartered in Hudson, Massachusetts, designs, manufactures and markets industry-leading semiconductor products including Alpha microprocessors and PCI chips for networking, bridging and multimedia, as well as low power Strong ARM microprocessors under license from Advanced RISC machines Ltd. Digital Semiconductor's PCI multimedia chips deliver quality compressed video to Windows-based desktop computers for applications such as MPEG-1 authoring, MPEG-1 video editing, H.320-based videoconferencing. Using a systems-oriented approach to both hardware and software design, Digital Semiconductor's multimedia chips deliver a more complete solution leveraging standard operationg system applications programming interfaces (APIs), host CPU performance and industry standard PCI-bus interfaces. More information about the complete line of Digital Semiconductor products visit our World Wide Web site: <http://www.digital.com/info/semiconductor>.

# TRIMEDIA - The Processor for PC-Consumer Multimedia

Selliah Rathnam, Gert Slavenburg

Philips Semiconductors

811 E. Arques Avenue, Sunnyvale, CA 94088

e-mail: selliah@trimedia.scs.philips.com

## ABSTRACT

TM-1 is the first in a family of programmable multimedia processors from the Trimedia product group of Philips Semiconductors. This PCI bus based "C" programmable processor has a high performance VLIW-CPU core with video and audio peripheral units designed to support the popular multimedia applications. TM-1 is designed to concurrently process video, audio, graphics, and communication data. The VLIW-CPU core is capable of executing a maximum of twenty seven operations per cycle; and the sustained execution rate is about five operations per cycle for the tuned applications. The audio unit easily handles different audio formats including the 16-bit stereo data. The video unit is capable of processing different YUV and RGB pixel formats with horizontal and vertical scaling and color space conversion.

The pixel image data in RGB format can be sent to the graphics device through PCI bus. TM-1 applications can range from low-cost, stand alone systems such as video phones to programmable, multipurpose plug-in cards for traditional computers.

## 1.0 INTRODUCTION

TM-1 is a building-block for high-performance multimedia applications that deal with high-quality video and audio. TM-1 easily implements popular multimedia standards such as MPEG-1 and MPEG-2, but its orientation around a powerful general-purpose CPU makes it capable of implementing a variety of multimedia algorithms, whether open or proprietary.

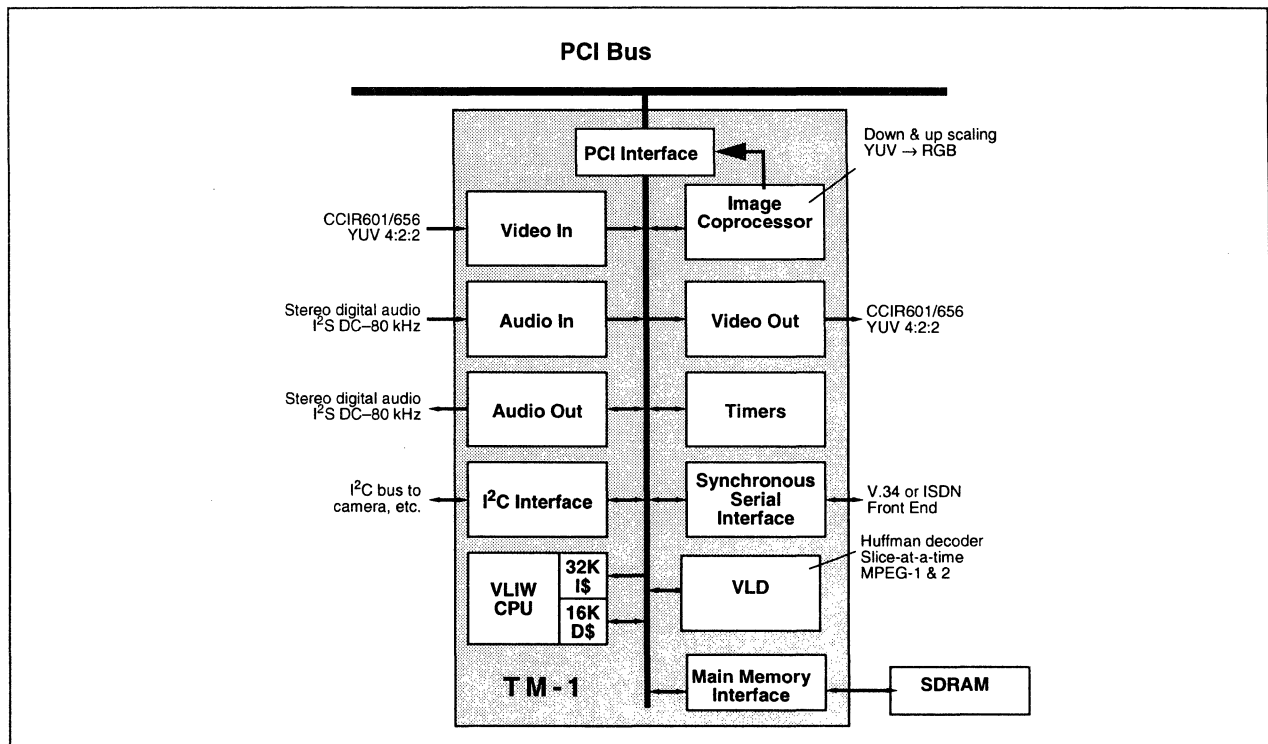
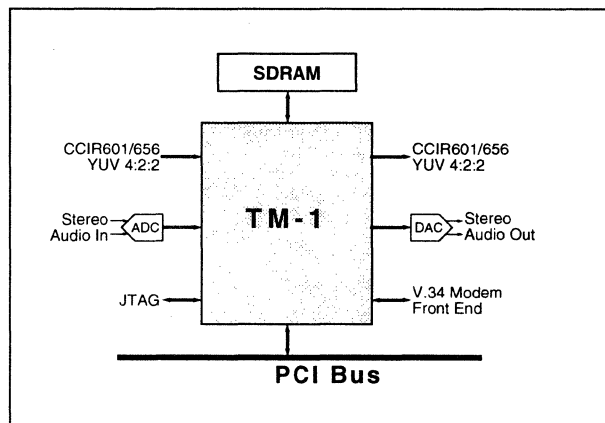


Figure 1. TM-1 block diagram.



**Figure 2. TM-1 system connections. A minimal TM-1 system requires few supporting components.**

More than just an integrated microprocessor with unusual peripherals, the TM-1 microprocessor is a fluid computer system controlled by a small real-time OS kernel that runs on the VLIW processor core. TM-1 contains a CPU, a high-bandwidth internal bus, and internal bus-mastering DMA peripherals.

TM-1 is the first member of a family of chips that will carry investments in software forward in time. Compatibility between family members is at the source-code level; binary compatibility between family members is not guaranteed. All family members, however, will be able to perform the most important multimedia functions, such as running MPEG-2 software.

Defining software compatibility at the source-code level gives Philips the freedom to strike the optimum balance between cost and performance for all the chips in the TM-1 family. Powerful compilers ensure that programmers seldomly need to resort to non-portable assembler programming. Programmers use TM-1's powerful low-level operations from source code; these DSP-like operations are invoked with a familiar function-call syntax. Trimedia also provides hand-coded and tuned multimedia libraries which can be used to increase the performance of the multimedia applications.

As the first member of the family, TM-1 is tailored for use in PC-based applications. Because it is based on a general-purpose CPU, TM-1 can serve as a multi-function PC enhancement vehicle. Typically, a PC must deal with multi-standard video and audio streams, and users desire both decompression and compression, if possible. While the CPU chips used in PCs are becoming capable of low-resolution real-time video decompression, high-quality video decompression—not to mention compression—is still out of reach. Further, users demand that their systems provide live video and audio without sacrificing the responsiveness of the system.

TM-1 enhances a PC system to provide real-time multimedia, and it does so with the advantages of a special-purpose, embedded solution—low cost and chip count—and the advantages of a general-purpose processor—reprogrammability. For PC applications, TM-1 far sur-

passes the capabilities of fixed-function multimedia chips.

Other Trimedia family members will have different sets of interfaces appropriate for their intended use. For example, a TM-1 chip for a cable-TV decoder box would eliminate the video-in interface.

## 2.0 TM-1 CHIP OVERVIEW

The key features of TM-1 are:

- A very powerful, general-purpose VLIW processor core that coordinates all on-chip activities. In addition to implementing the non-trivial parts of multimedia algorithms, this processor runs a small real-time operating system that is driven by interrupts from the other units.
- DMA-driven multimedia input/output units that operate independently and that properly format data to make processing efficient.
- DMA-driven multimedia coprocessors that operate independently and perform operations specific to important multimedia algorithms.
- A high-performance bus and memory system that provides communication between TM-1's processing units.

Figure 1 shows a block diagram of the TM-1 chip. The bulk of a TM-1 system consists of the TM-1 microprocessor itself, a block of synchronous DRAM (SDRAM), and minimal external circuitry to interface to the incoming and/or outgoing multimedia data streams. TM-1 can gluelessly interface to the standard PCI bus for personal-computer-based applications; thus, TM-1 can be placed directly on the PC mainboard or on a plug-in card.

Figure 2 shows a possible TM-1 system application. A video-input stream, if present, might come directly from a CCIR 601-compliant digital video camera chip in YUV 4:2:2 format; the interface is glueless in this case. A non-standard camera chip can be connected via a video decoder chip (such as the Philips SAA7111). A CCIR 601 output video stream is provided directly from the TM-1 to drive a dedicated video monitor. Stereo audio input and output require external ADC and DAC support. The operation of the video and audio interface units is highly customizable through programmable parameters.

The glueless PCI interface allows the TM-1 to display video via a host PC's video card and to play audio via a host PC's sound hardware. The Image Coprocessor provides display support for live video in an arbitrary number of arbitrarily overlapped windows.

Finally, the V.34 interface requires only an external modem front-end chip and phone line interface to provide remote communication support. The modem can be used to connect TM-1-based systems for video phone or video conferencing applications, or it can be used for general-purpose data communication in PC systems.

### 3.0 BRIEF EXAMPLES OF OPERATION

The key to understanding TM-1 operation is observing that the CPU and peripherals are time-shared and that communication between units is through SDRAM memory. The CPU switches from one task to the next; first it decompresses a video frame, then it decompresses a slice of the audio stream, then back to video, etc. As necessary, the CPU issues commands to the peripheral units to orchestrate their operation.

The TM-1 CPU can enlist the ICP and video-in units to help with some of the straightforward, tedious tasks associated with video processing. The function of these units is programmable. For example, some video streams are—or need to be—scaled horizontally, so these units can handle the most common cases of horizontal down- and up-scaling without intervention from the TM-1 CPU.

#### 3.1 Video Decompression in a PC

A typical mode of operation for a TM-1 system is to serve as a video-decompression engine on a PCI card in a PC. In this case, the PC doesn't know the TM-1 has a powerful, general-purpose CPU; rather, the PC just treats the hardware on the PCI card as a "black-box" engine.

Video decompression begins when the PC operating system hands the TM-1 a pointer to compressed video data in the PC's memory (the details of the communication protocol are typically handled by a software driver installed in the PC's operating system).

The TM-1 CPU fetches data from the compressed video stream via the PCI bus, decompresses frames from the video stream, and places them into local SDRAM. Decompression may be aided by the VLD (variable-length decoder) unit, which implements Huffman decoding and is controlled by the TM-1 CPU.

When a frame is ready for display, the TM-1 CPU gives the ICP (image coprocessor) a display command. The ICP then autonomously fetches the decompressed frame data from SDRAM and transfers it over the PCI bus to the frame buffer in the PC's video display card (or the frame buffer in PC system memory if the PC uses a UMA (Unified Memory Architecture) frame buffer). The ICP accommodates arbitrary window size, position, and overlaps.

#### 3.2 Video Compression

Another typical application for TM-1 is in video compression. In this case, uncompressed video is usually supplied directly to the TM-1 system via the video-in unit. A camera chip connected directly to the video-in unit supplies YUV data in eight-bit, 4:2:2 format. The video-in unit takes care of sampling the data from the camera chip and demultiplexing the raw video to SDRAM in three separate areas, one each for Y, U, and V.

When a complete video frame has been read from the camera chip by the video-in unit, it interrupts the TM-1 CPU. The CPU compresses the video data in software (using a set of powerful data-parallel operations) and writes the compressed data to a separate area of

SDRAM.

The compressed video data can now be disposed of in any of several ways. It can be sent to a host system over the PCI bus for archival on local mass storage, or the host can transfer the compressed video over a network, such as ISDN. The data can also be sent to a remote system using the integrated V.34 interface to create, for example, a video phone or video conferencing system.

Since the powerful, general-purpose TM-1 CPU is available, the compressed data can be encrypted before being transferred for security.

### 4.0 VLIW CORE AND PERIPHERAL UNITS

#### 4.1 VLIW Processor Core

The heart of TM-1 is its powerful 32-bit CPU core. The CPU implements a 32-bit linear address space and 128, fully general-purpose 32-bit registers. The registers are not separated into banks; any operation can use any register for any operand.

The core uses a VLIW instruction-set architecture and is fully general-purpose. TM-1 uses a VLIW instruction length that allows up to five simultaneous operations to be issued. These operations can target any five of the 27

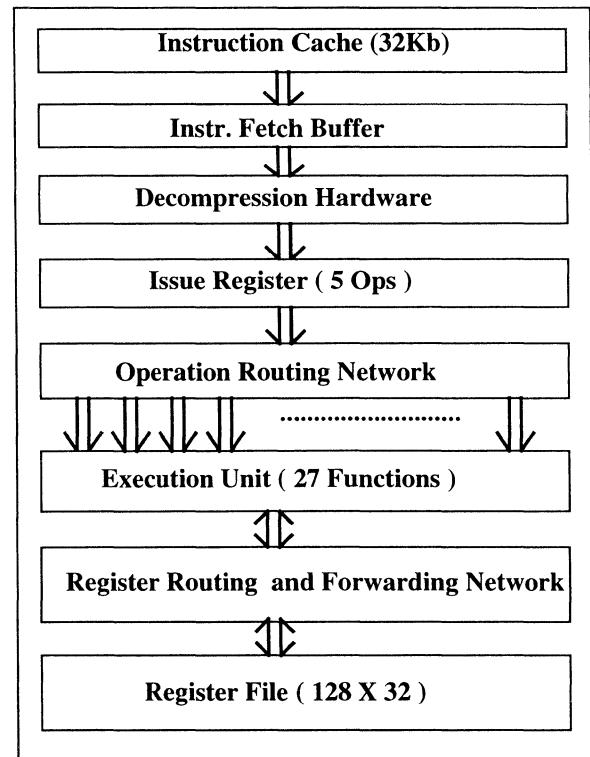


Figure 3. VLIW Processor Core and Instruction Cache.



functional units in the CPU, including integer and floating-point arithmetic units and data-parallel DSP-like units.

Although the processor core runs a tiny real-time operating system to coordinate all activities in the TM-1 system, the processor core is not intended for true general-purpose use as the only CPU in a computer system. For example, the processor core does not implement virtual memory address translation, an essential feature in a general-purpose computer system.

TM-1 uses a VLIW architecture to maximize processor throughput at the lowest possible cost. VLIW architectures have performance exceeding that of superscalar general-purpose CPUs without the extreme complexity of a superscalar implementation. The hardware saved by eliminating superscalar logic reduces cost and allows the integration of multimedia-specific features that enhance the power of the processor core.

The TM-1 operation set includes all traditional microprocessor operations. In addition, multimedia-specific operations are included that dramatically accelerate standard video compression and decompression algorithms. As just one of the five operations issued in a single TM-1 instruction, a single special or "custom" operation can implement up to 11 traditional microprocessor operations. Multimedia-specific operations combined with the VLIW architecture result in tremendous throughput for multimedia applications.

#### 4.2 Internal "Data Highway" Bus

The internal data bus connects all internal blocks together and provides access to internal control registers (in each on-chip peripheral units), external SDRAM, and the external PCI bus. The internal bus consists of separate 32-bit data and address buses, and transactions on the bus use a block-transfer protocol. Peripherals can be masters or slaves on the bus.

Access to the internal bus is controlled by a central arbiter, which has a request line from each potential bus master. The arbiter is configurable in a number of different modes so that the arbitration algorithm can be tailored for different applications. Peripheral units make requests to the arbiter for bus access, and depending on the arbitration mode, bus bandwidth is allocated to the units in different amounts. Each mode allocates bandwidth differently, but each mode guarantees each unit a minimum bandwidth and maximum service latency. All unused bandwidth is allocated to the TM-1 CPU.

The bus allocation mechanism is one of the features of TM-1 that makes it a true real-time system instead of just a highly integrated microprocessor with unusual peripherals.

#### 4.3 Memory and Cache Units

TM-1's memory hierarchy satisfies the low cost and high bandwidth requirement of multimedia markets. Since multimedia video streams can require relatively large temporary storage, a significant amount of DRAM

is required.

TM-1 has a glueless interface with synchronous DRAM (SDRAM) or synchronous graphics RAM (SGRAM), which provide higher bandwidth than the standard DRAM. As the SDRAM has been supported by major DRAM vendors, the competition among those vendors will keep the SDRAM price in par with that of the standard DRAM. TM-1's DRAM memory size can range from 2Mbytes to 64 Mbytes.

The TM-1 CPU core is supported by separate 16-KB data and 32-KB instruction caches. The data cache is dual-ported in order to allow two simultaneous load/store accesses, and both caches are eight-way set-associative with a 64-byte block size.

#### 4.4 Video-In Unit

The video-in unit interfaces directly to any CCIR 601/656-compliant device that outputs eight-bit parallel, 4:2:2 YUV time-multiplexed data. Such devices include direct digital camera systems, which can connect gluelessly to TM-1 or through the standard CCIR 656 connector with only the addition of ECL level converters. Non-CCIR-compliant devices can use a digital decoder chip, such as the Philips SAA7111, to interface to TM-1. Older front ends with a 16-bit interface can connect with a small amount of glue logic.

The video-in unit demultiplexes the captured YUV data before writing it into local TM-1 SDRAM. Separate data structures are maintained for Y, U, and V.

The video-in unit can be programmed to perform on-the-fly horizontal resolution subsampling by a factor of two if needed. Many camera systems capture a 640-pixel/line or 720-pixel/line image; with subsampling, direct conversion to a 320-pixel/line or a 360-pixel/line image can be performed with no CPU intervention. Further, if subsampling is required eventually, performing this function during data capture reduces initial storage requirements.

#### 4.5 Video-Out Unit

The video-out unit essentially performs the inverse function of the video-in unit. Video-out generates an eight-bit, multiplexed YUV data stream by gathering bits from the separate Y, U, and V data structures in SDRAM. While generating the multiplexed stream, the video-out unit can also up-scale horizontally by a factor of two to convert from CIF to native CCIR resolution.

Since the video-out unit likely drives a separate video monitor—not the PC's video screen—the PC itself cannot be used to generate the graphics and text of a user interface. To remedy this, the video-out unit can generate graphics overlays in a limited number of configurations.

#### 4.6 Image Coprocessor (ICP)

The image coprocessor (ICP) is used for several purposes to off-load tasks from the TM-1 CPU, such as copying an image from SDRAM to the host's video frame buffer. Although these tasks can be easily performed by the CPU, they are a poor use of the relatively

expensive CPU resource. When performed in parallel by the ICP, these tasks are performed efficiently by simple hardware, which allows the CPU to continue with more complex tasks.

The ICP can operate as either a memory-to-memory or a memory-to-PCI coprocessor device.

In memory-to-memory mode, the ICP can perform either horizontal or vertical image filtering and resizing. The ICP implements 32 FIR filters of five adjacent pixel input values. The filter coefficients are fully programmable, and the position of the output pixel in the output raster determines which of the 32 FIR filters is applied to generate that output pixel value. Thus, the output raster is on a 32-times finer grid than the input raster. The filtering is done in either the horizontal or vertical direction but not both. Two applications of the ICP are required to filter and scale in both directions.

In memory-to-PCI mode, the ICP can perform horizontal resizing followed by color-space conversion. For example, assume an  $n \times m$  pixel array is to be displayed in a window on the PC video screen while the PC is running a graphical user interface. The first step (if necessary) would use the ICP in memory-to-memory mode to perform a vertical resizing. The second step would use the ICP in memory-to-PCI mode to perform a horizontal resizing (if necessary) and colorspace conversion from YUV to RGB.

While sending the final, resampled and converted pixels over the PCI bus to the video frame buffer, the ICP uses a full, per-pixel occlusion bit mask—accessed in destination coordinates—to determine which pixels are actually stored in the frame buffer for display. Condi-

tioning the transfer with the bit mask allows TM-1 to accommodate an arbitrary arrangement of overlapping windows on the PC video screen.

Figure 3 illustrates a possible display situation and the data structures in SDRAM that support the ICP's operation. On the left in Figure 3, the PC's video screen has four overlapping windows. Two, Image 1 and Image 2, are being used to display video generated by TM-1.

The right side of Figure 3 shows a conceptual view of SDRAM contents. Two data structures are present, one for Image 1 and the other for Image 2. Figure 3 represents a point in time during which the ICP is displaying Image 2.

When the ICP is displaying an image (i.e., copying it from SDRAM to a frame buffer), it maintains four pointers to the data structures in SDRAM. Three pointers locate the Y, U, and V data arrays, and the fourth locates the per-pixel occlusion bit map. The Y, U, and V arrays are indexed by source coordinates while the occlusion bit map is accessed with screen coordinates.

As the ICP generates pixels for display, it performs horizontal scaling and colorspace conversion. The final RGB pixel value is then copied to the destination address in the screen's frame buffer only if the corresponding bit in the occlusion bit map is a one.

As shown in the conceptual diagram, the occlusion bit map has a pattern of 1s and 0s that corresponds to the shape of the visible area of the destination window in the frame buffer. When the arrangement of windows on the PC screen is changed, modifications to the occlusion bit maps may be necessary.

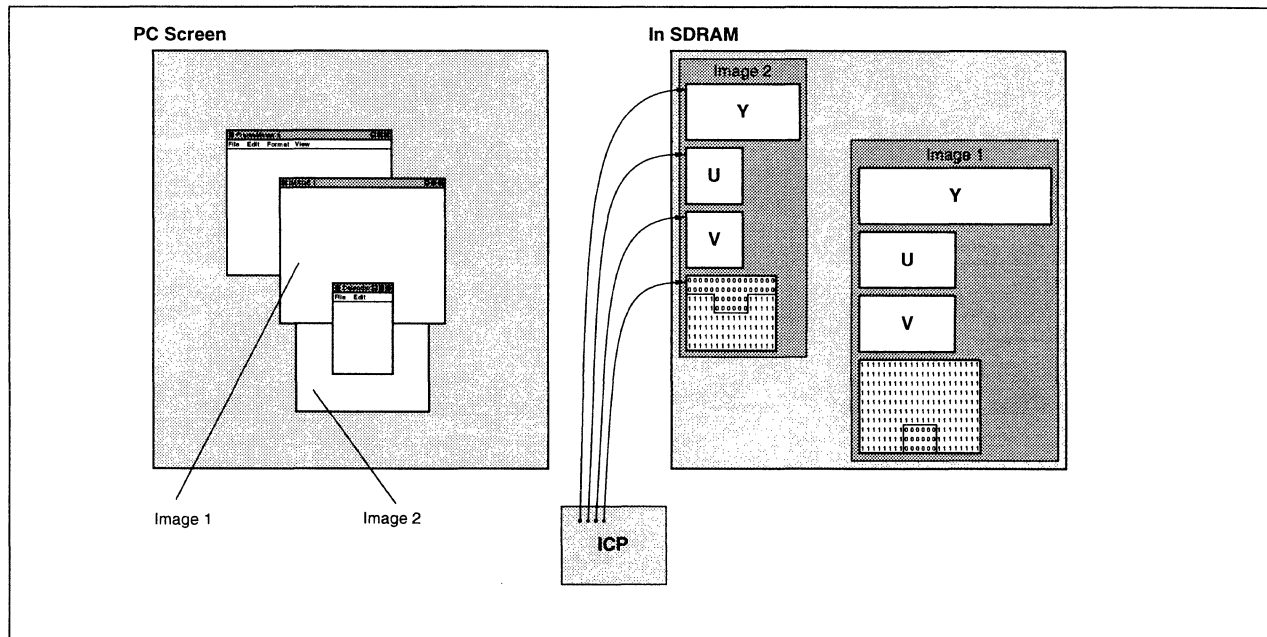


Figure 4. ICP operation. Windows on the PC screen and data structures in SDRAM for two live video windows.

It is important to note that there is no preset limit on the number and sizes of windows that can be handled by the ICP. The only limit is the available bandwidth. Thus, the ICP can handle a few large windows or many small windows. The ICP can sustain a transfer rate of 50 megapixels per second, which is more than enough to saturate PCI when transferring images to video frame buffers.

ICP has a micro-programmable engine. All ICP operations such as filtering, scaling and color space conversions and their formats are programmable. The ICP's micro programs loads itself from the SDRAM memory.

#### 4.7 Variable-Length Decoder (VLD)

The variable-length decoder (VLD) is included to relieve the TM-1 CPU of the task of decoding Huffman-encoded video data streams. It can be used to help decode MPEG-1 and MPEG-2 video streams.

The VLD is a memory-to-memory coprocessor. The TM-1 CPU hands the VLD a pointer to a Huffman-encoded bit stream, and the VLD produces a tokenized bit stream that is very convenient for the TM-1 image decompression software to use. The format of the output token stream is optimized for the MPEG-2 decompression software so that communication between the CPU and VLD is minimized.

As with the other processing-intensive coprocessors, the VLD is included mainly to relieve the CPU of a task that wastes its performance potential. When dealing with the high bit rates of MPEG-2 data streams, too much of the CPU's time is devoted to this task, which prevents its special capabilities from being used.

#### 4.8 Audio-In and Audio-Out Units

The audio-in and audio-out units are similar to the video units. They connect to most serial ADC and DAC chips, and are programmable enough to handle most reasonable protocols. These units can transfer MSB or LSB first and left or right channel first.

The sampling clock is driven by TM-1 and is software programmable within a wide range from DC to 80 kHz with a resolution of 0.02 Hz. The clock circuit allows the programmer subtle control over the sampling frequency so that audio and video synchronization can be achieved in any system configuration. When changing the frequency, the instantaneous phase does not change, which allows frequency manipulation without introducing dis-

ortion.

As with the video units, the audio-in and audio-out units buffer incoming and outgoing audio data in SDRAM. The audio-in unit buffers samples in either eight- or 16-bit format, mono or stereo. The audio-out unit simply transfers sample data from memory to the external DAC; any manipulation of sound data is performed by the TM-1 CPU since this processing will require at most a few percent of the CPU resource.

### 5.0 PCI BUS INTERFACE UNIT (BIU)

TM1 is capable of operating either as a main CPU in a stand alone system or as a PCI peripheral device in a desktop PC (Figure 5).

This unit connects the internal Data Highway Bus to an external PCI bus. It has a PCI master to initiate memory read/write cycles for TM-1-CPU requested read/write transactions including burst read/write DMA transactions. The PCI target within the BIU responds to the transactions initiated by external PCI master devices to read/write the TM-1's memory space, and it satisfies their requests. External devices can access the TM-1's MMIO registers through this unit.

The ICP unit has a direct connection to the BIU unit in order to transfer the pixel image data efficiently from TM-1 to the graphics device or host memory through the PCI bus.

#### 5.1 PCI INTERFACE AS AN INITIATOR

Three classes of operations invoked by TM-1 cause the PCI interface to act as a PCI initiator:

- Transparent, single-word (or smaller) transactions caused by DSPCPU loads and stores to the PCI address aperture.
- Explicitly programmed single-word PCI-bus I/O transactions.
- Explicitly programmed multi-word DMA transactions.

#### 5.2 DSPCPU Single-Word Loads/Stores

From the point of view of programs executed by TM-1's DSPCPU, there are three apertures into TM-1's 4-GB

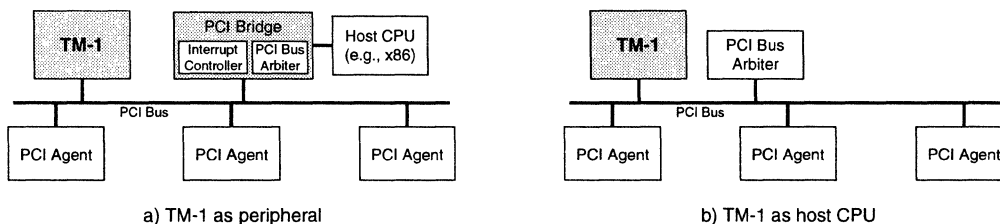


Figure 5. Two typical system implementations. (a) shows TM-1 as a PCI peripheral in a desktop PC. (b) shows an embedded system with TM-1 as the host CPU.

memory address space:

- SDRAM space (0.5 to 64 MB in size).
- MMIO space (2 MB in size).
- PCI space.

MMIO registers control the positions and extent of the address-space apertures. The SDRAM aperture begins at the address specified in the MMIO register DRAM\_BASE and extends upward to the address in the DRAM\_LIMIT register. The 2-MB MMIO aperture begins at the address in MMIO\_BASE. All addresses that fall outside these two apertures are assumed to be part of the PCI address aperture. References by DSPCPU loads and stores to the PCI aperture are reflected to external PCI devices by the coordinated action of the data cache and PCI interface.

When a DSPCPU load or store targets the PCI aperture (i.e., neither of the other two apertures), the DSPCPU's data cache automatically carries out a special sequence of events. The data cache writes to the PCI\_ADR and (if the DSPCPU operation is a store) the PCI\_DATA registers in the PCI interface and asserts (load) or deasserts (store) the internal signal pci\_read\_operation.

While the PCI interface executes the PCI bus transaction, the DSPCPU is held in the stall state by the data cache. When the PCI interface has completed the transaction, it asserts the internal signal pci\_ready.

When pci\_ready is asserted, the data cache finishes the original DSPCPU operation by reading data from the PCI\_DATA register (if the DSPCPU operation was a load) and releasing the DSPCPU from the stall state.

### 5.3 I/O Operations

Explicit programming by DSPCPU software is the way to perform transactions to PCI I/O space. DSPCPU software writes three MMIO registers in the following sequence:

1. The IO\_ADR register.
2. The IO\_DATA register (if PCI operation is a write).
3. The IO\_CTL register (controls direction of data movement and which bytes participate).

The PCI interface starts the PCI-bus I/O transaction when software writes to IO\_CTL. The interface can raise a DSPCPU interrupt at the completion of the I/O transaction or the DSPCPU can poll the appropriate status bit.

### 5.4 DMA Operations

The PCI interface can operate as an autonomous DMA engine, executing block-transfer operations at maximum PCI bandwidth. As with I/O and configuration operations, DSPCPU software explicitly programs DMA operations.

#### **General-purpose DMA**

For DMA between SDRAM and PCI, DSPCPU soft-

ware writes three MMIO registers in the following sequence:

1. The SRC\_ADR and DEST\_ADR registers.
2. The DMA\_CTL register (controls direction of data movement and amount of data transferred).

The PCI interface begins the PCI-bus transactions when software writes to DMA\_CTL. As with the I/O and configuration operations, the BIU\_STATUS and BIU\_CTL registers monitor the status of the operation and control interrupt signalling.

#### **Image-Coprocessor DMA**

The PCI interface also executes DMA transactions for the Image Coprocessor (ICP). The ICP performs rapid post-processing of pixel data and writes it at DMA speed to an external video frame buffer. The ICP cannot perform PCI read transactions.

The DMA transactions are considered as background transactions. To reduce the latency of the single word read/write transactions on the PCI bus, the BIU interleaves the burst read/write DMA cycles with single word read/write transactions.

## 6.0 APPLICATIONS

TM-1 has the potential to be used in many multimedia applications and only few of them are discussed.

### 6.1 Video Teleconferencing/Digital White Board

Businesses are increasingly turning towards interactive computing as a means of becoming more efficient. Collaborative computing, for instance, involves sharing applications amongst multiple personal computers and multipoint video teleconferencing.

TM-1 is a single chip video teleconferencing solution that runs all current video codecs across all common transport mechanisms. This may also include H.324 (POTS), H.320 (ISDN) and H.323 (LAN).

### 6.2 Multimedia Card for Consumer Multimedia Applications

The achievement of true computer based realism is only possible with a fully integrated approach to multimedia -- one that permits the smooth flow of audio, video, graphics and communications. Today's computer user wants a highly interactive and realistic experience. The Trimedia processor makes this possible.

TM-1 is a low-cost, programmable processor for the consumer multimedia market. This product provides the additional processing power required for a true-to-life computer based experience. The Trimedia processor concurrently processes multiple data types including audio, video, graphics and communications. The first version of this chip, designated TM-1, is targeted for the PC market.

## 7.0 SUMMARY

The TM-1 is the first programmable multimedia processor from the Trimedia division of the Philips Semiconductors. The PCI based TM-1 has high performance VLIW CPU core, efficient 'C' compiler with multimedia library functions, glueless logic to high-bandwidth SDRAM, standard PCI bus interface, and standard interfaces to video and audio stream that make the TM-1 the next generation multimedia processor for stand-alone systems such as the video phone, video conferencing system and plug-in multimedia cards for the PC systems.

## 8.0 REFERENCES

- [1] J. Labrousse, G. A. Slavenburg. "A 50MHz Microprocessor with a VLIW Architecture." ISSCC, 1990.
- [2] J. Labrousse, G. A. Slavenburg. "CREATE-LIFE: A Design System for High Performances VLSI Circuits" ICCD-88. 1988.
- [3] J. Labrousse, G. A. Slavenburg. "CREATE-LIFE: A Modular Design Approach for High Performances ASIC's." Comcon Conference 1990.
- [4] Brian Case. "Philips Hopes to Displace DSPs with VLIW" Microprocessor Report, December 5, 1994.
- [5] Brian Case. "First Trimedia Chip Boards PCI Bus." Microprocessor Report, November 1995.
- [6] Gert Slavenburg. "The Trimedia VLIW-Based PCI Multimedia Processor" In Microprocessor Forum, October, 1995.
- [7] A.S. Huang, G. Slavenburg, J.P. Shen. "Speculative Disambiguation: A compilation Technique for Dynamic Memory Disambiguation". In 21st Annual International Symposium on Computer Architecture, April, 1994.
- [8] R.P. Colwell, R.P. Nix, J.J O'Donnell, D.B. Papworth, P.K. Rodman. "A VLIW Architecture for a Trace Scheduling Compiler." Proc. of ASPLOS II. October, 1987.
- [9] J.A. Fisher. "Trace Scheduling: A Technique for Global Microcode Compaction." IEEE Trans on Computers. July 1981.
- [10] P.Y.T. Hsu and E.S. Davidson. "Highly Concurrent Scalar Processing." Proc. of the 13th Symposium on Computer Architecture, 1986

# **Multimedia Bandwidth Issues over PCI**

**Author: Giri Venkat**

## **Abstract**

With the increasing integration of multimedia processors into the mainstream PC environment, comes an increasing demand on the system's peripheral bus. This paper will analyze some of the issues relating to multimedia over the PCI bus. Topics such as MPEG-2, 3D graphics and full motion video will be explored as they relate to PCI. Following a brief overview of multimedia technologies and their bandwidth requirements will be a discussion of a mezzanine PCI architecture that will ease the bandwidth demands over the main PCI bus. Finally, a comparison of a standard PCI system vs. a mezzanine implementation will be presented to demonstrate the advantages of a mezzanine architecture for multimedia applications.

# Board Maker's Roundtable

*Steve Cooper, I-Bus*

The bus/board marketplace continues to be an exciting, growing marketplace that provides:

- a standard way for system designers to build systems from higher-level building blocks available from multiple companies (boards versus components)
- the rapid availability and infusion of new technologies into systems designs

The advent of the PCI bus is playing a significant role in the ongoing evolution of bus structures. As a number of other popular buses (Multibus, STD, VME) grow older, various forms of the PCI bus are stepping in to solve solutions problems.

One version of the PCI bus, known as the PICMG standard for passive backplane PCs, is among the fastest growing new bus structures available. This standard was approved in October 1994 and already has become a dominant standard in many market sectors. The future of this new bus can be seen by observing that the latest technologies are appearing on this bus structure prior to other bus structures. For example, the first bus structure (other than the office PC motherboard) to host Intel's latest P6 Pentium Pro CPU chips is this PICMG-compliant PCI/ISA bus.

Another version of the PCI bus, known as the CompactPCI standard, is now poised to move into more embedded applications. This bus structure was recently completed by the PICMG standardization group, and the first products from a variety of vendors are beginning to appear.

Compared with many computer components, bus structures tend to be relatively long-lived. Many bus structures have remained viable for over 20 years, continually finding ways to accommodate to the industries ever-changing components. Thus, the development and evolution of these new PCI-based buses marks a significant event for the bus/board suppliers as well as the marketplace that use them.

# **Bus-to-Bus Connections**

*Stephan Ohr, Computer Design*

There is a wealth of peripherals like digital cameras, scanners, and data acquisition equipment that take advantage of PCI's high data transfer rates. Since the variety of peripherals for Apple computers and digital signal processors is not as great, bridge cards and devices are emerging which link PCI peripherals with other bus topologies. Presenters in this session offer insights into the process of interfacing PCI to the Apple NuBus, the P1394 video bus, even the IBM Serial Storage Architecture (SSA).



**COROLLA**

# **PCI and Multiprocessing**

**George P. White**

**PCI Spring Developers' Conference**

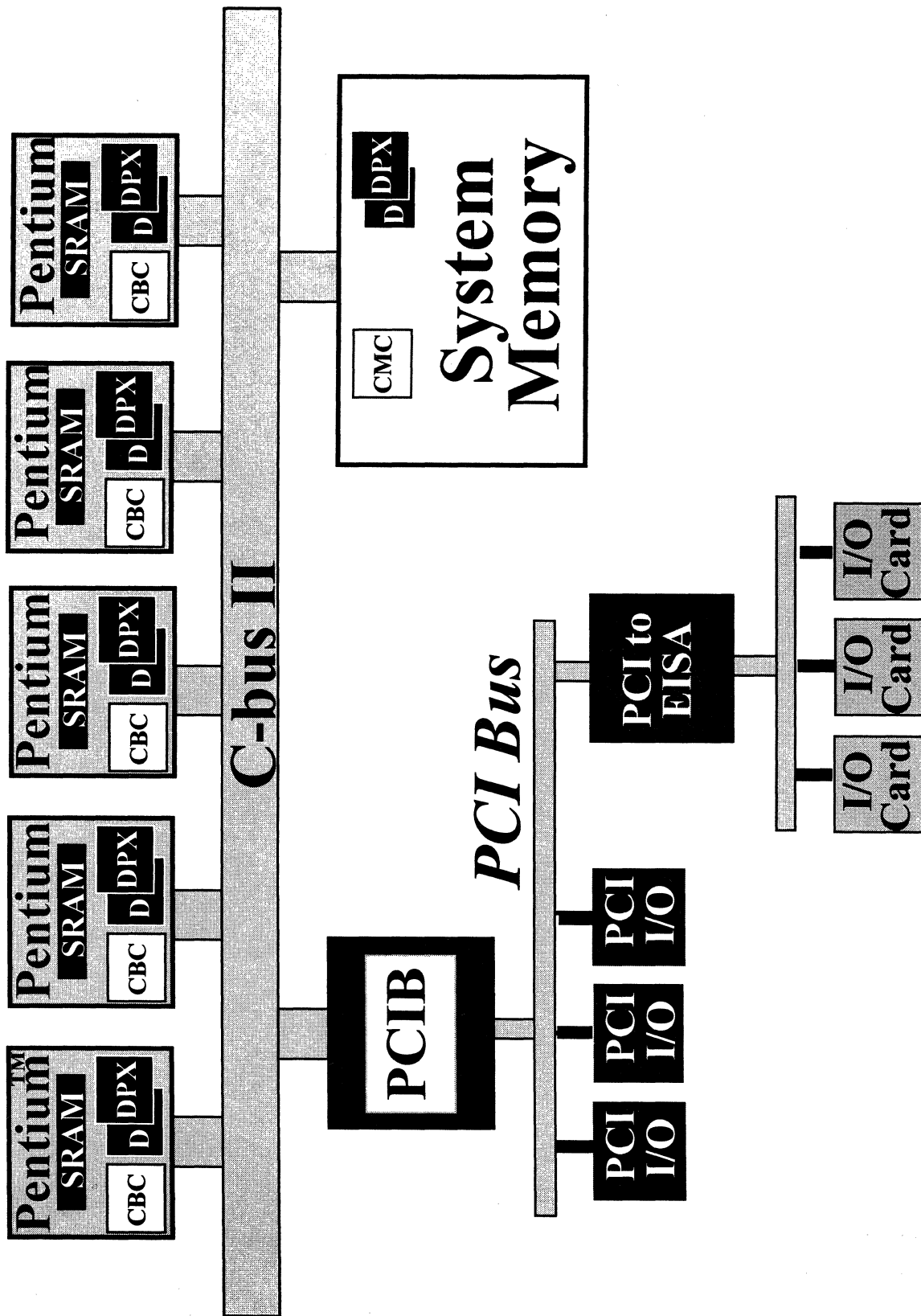
**April 29 - May 3, 1996**

**San Jose, CA**

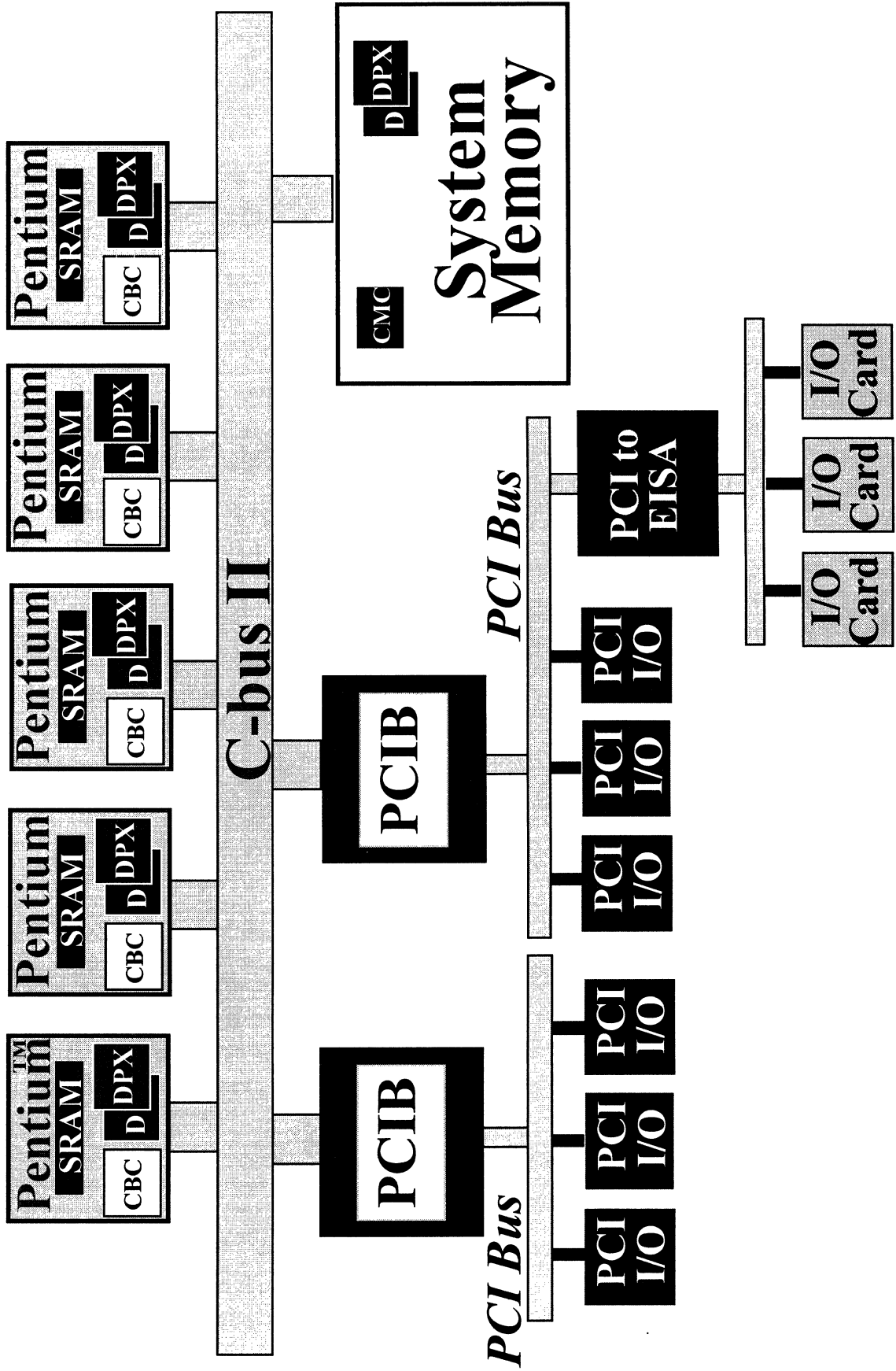
# **Multiple PCI Bridges for Specialized Multiprocessor Bus**

- ◆ **Eight Pentiums supported**
- ◆ **Two peer PCI buses supported**
- ◆ **One chip PCI interface**

# C-bus II Architecture



# C-bus II: Dual PCI



# C-bus II Features

- ◆ **400 MB/sec. bandwidth**
- ◆ **32 GB physical memory space**
- ◆ **64-bit multiplexed address and data**
- ◆ **ECC protection on bus transactions**
- ◆ **Multiple I/O buses supported**
- ◆ **Simple control protocol related to NuBus**

# Innovative Technology

## *C-bus II*

**GTL signaling**

**ECC on bus**

**50 MHz**

**64 bits**

**Multiplexed  
address/data**

## *P6 bus*

**GTL + signaling**

**ECC on bus**

**66 MHz**

**64 bits**

**Pipelined bus**

# C-bus II Signal Summary

<b>Signal</b>	<b>No.</b>	<b>Description</b>
<b>CBCLK</b>	<b>1</b>	<b>System Clock (50 MHz)</b>
<b>CD [63..0]</b>	<b>64</b>	<b>Multiplexed Address/Data</b>
<b>CDE [7..0]</b>	<b>8</b>	<b>Address/Data Error Correction Code (ECC)</b>
<b>STRT#</b>	<b>1</b>	<b>Transfer Start</b>
<b>TM [3..0]#</b>	<b>4</b>	<b>Transfer Mode (including parity)</b>
<b>ACK#</b>	<b>1</b>	<b>Transfer Acknowledge</b>
<b>CID [3..0]</b>	<b>4</b>	<b>Geographical Slot Identification</b>
<b>CARB [3..0]</b>	<b>4</b>	<b>Distributed Arbitration</b>
<b>LOCKCB#</b>	<b>1</b>	<b>Bus Lock</b>
<b>FAULT#</b>	<b>1</b>	<b>Hardware Fault Indicator</b>
<b>CRST#</b>	<b>1</b>	<b>System Reset</b>
<b>LED#</b>	<b>1</b>	<b>Light Emitting</b>
<b>Total</b>	<b>91</b>	<b>Active Signals</b>

# Cached Transactions

## Clock



## Read



Read cache line

## Eviction



Evict cache line

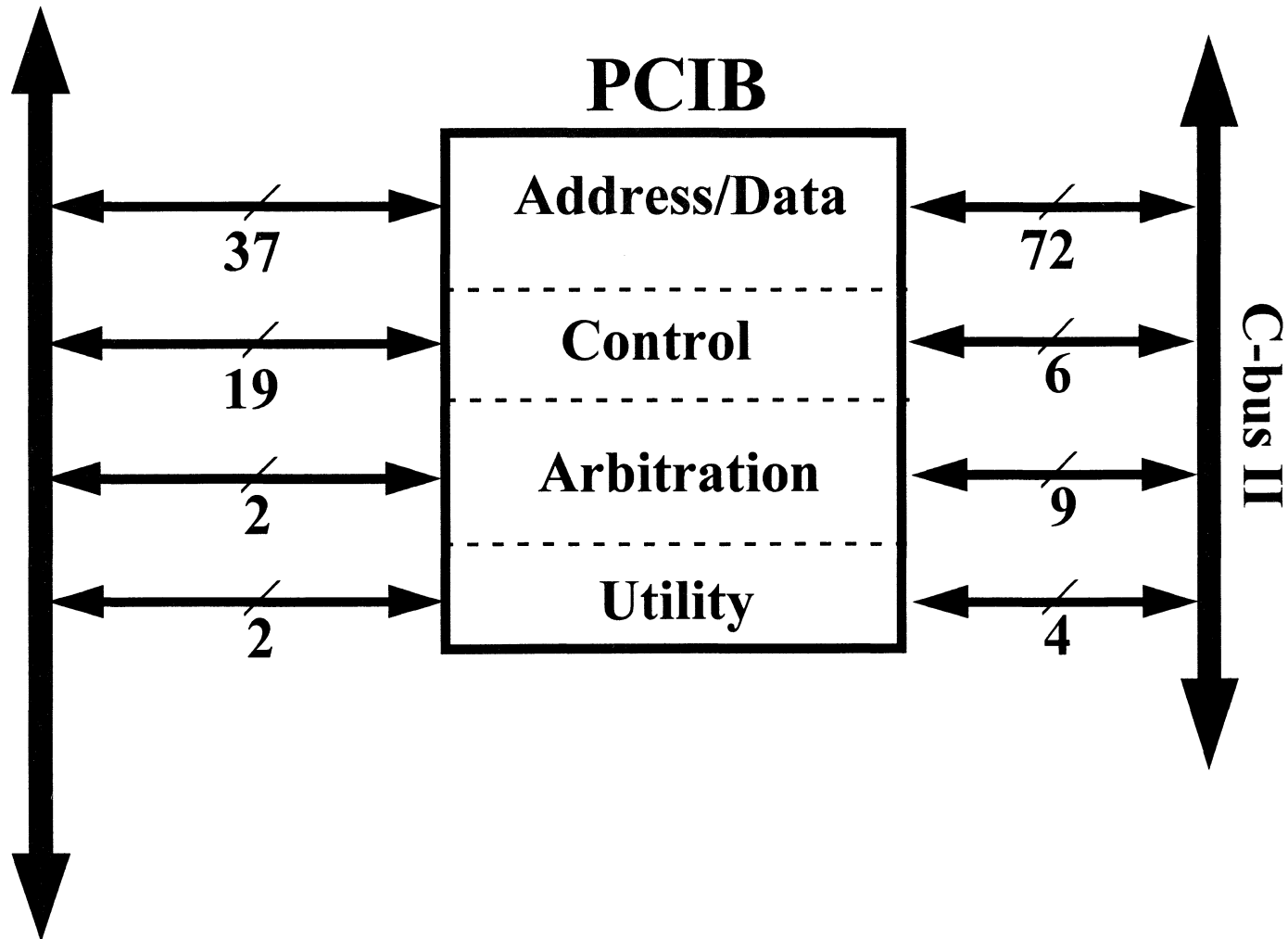
Read cache line



# PCI Bridge (PCIB)

- ◆ **Direct connection to C-bus II**
- ◆ **Direct connection to PCI bus**
- ◆ **Single chip solution**
- ◆ **DMA cache with prefetch buffers for maximum burst rate support**
- ◆ **Contains PCI central resource functions**
- ◆ **Handles distributed interrupts**
- ◆ **240 pin, Plastic Quad Flat Pack (PQFP)**

# PCIB (PCI to C-bus II) ASIC



# Caching PCI Interface

*Performance optimized by cache coherent interface*

- ◆ **Eight 32 byte cache lines**
- ◆ **Fully associative cache**
- ◆ **Leap-frogging read-ahead buffers**

# Support for Peer Bridges

- ◆ **Architecturally supports four PCI bridges, now shipping with two**
- ◆ **Separate address space for each bus mapped into system address space**
- ◆ **Supported by Corollary HAL for Windows NT**

# Who's Using this Technology ?

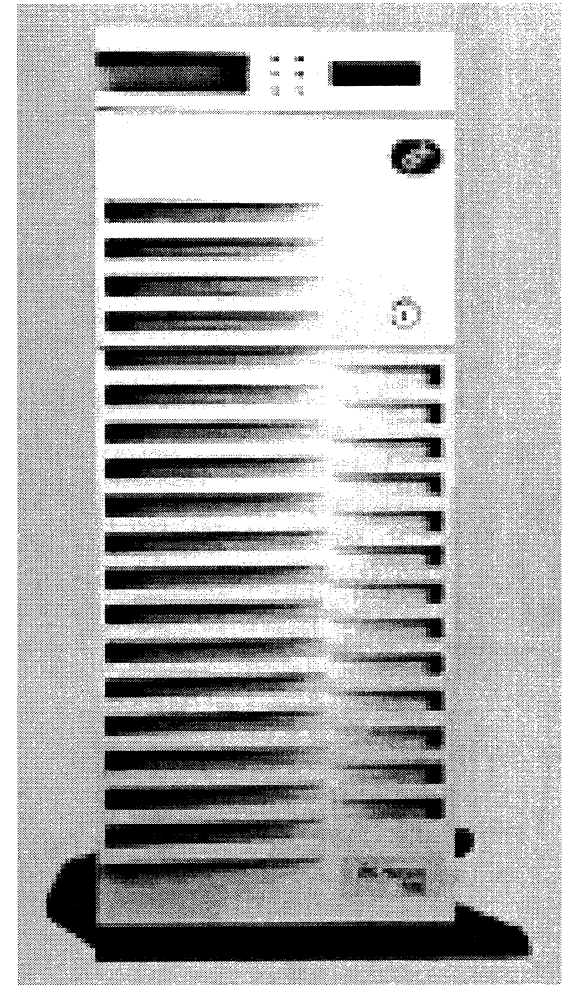
CPU's

<b>Chen</b>	<i>Chen 1000</i>	<b>8</b>
<b>DG</b>	<i>AViiON 4700, 4800, 5800</i>	<b>2,4,8</b>
<b>Fujitsu</b>	<i>FM-Server 7500SV</i>	<b>8</b>
<b>Hitachi</b>	<i>3100LP</i>	<b>8</b>
<b>IBM</b>	<i>PC Server 720</i>	<b>6</b>
<b>Intergraph</b>	<i>InterServe MP6 Servers</i>	<b>6</b>
<b>NEC</b>	<i>Express5800/170</i>	<b>6</b>
<b>Olivetti</b>	<i>SNX 400/RS Systems</i>	<b>4</b>
<b>Samsung</b>	<i>SSM 500</i>	<b>4</b>

COROLLARY

# Example - PC Company Moving Up

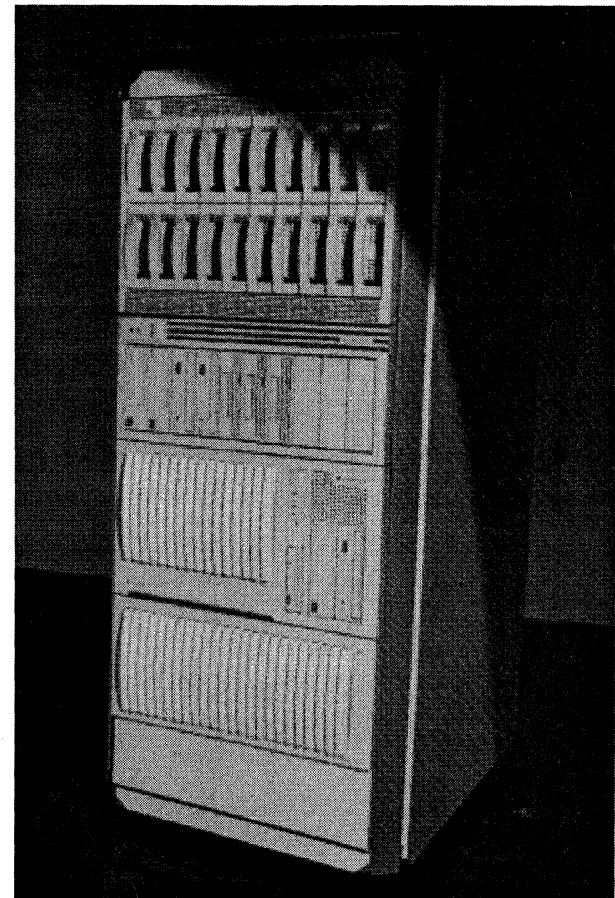
- ◆ IBM's first symmetric MP PC server
- ◆ Packaged like PC server
- ◆ Six CPUs
- ◆ One PCI bus supported



# Midrange Computer Company Getting More Open

*DG AViiON Models 4700, 4800, 5800*

- ◆ Packaged like mini computer
- ◆ 8 CPUs, 2 megabyte cache each
- ◆ Two PCI buses supported



COROLLARY

# Summary

*C-bus II optimized for cache-coherent multiprocessing Pentium design supports multiple peer PCI buses for high I/O throughput.*

*True caching and read-ahead for optimum performance.*



# **1394 and PCI**

**Larry Blackledge**

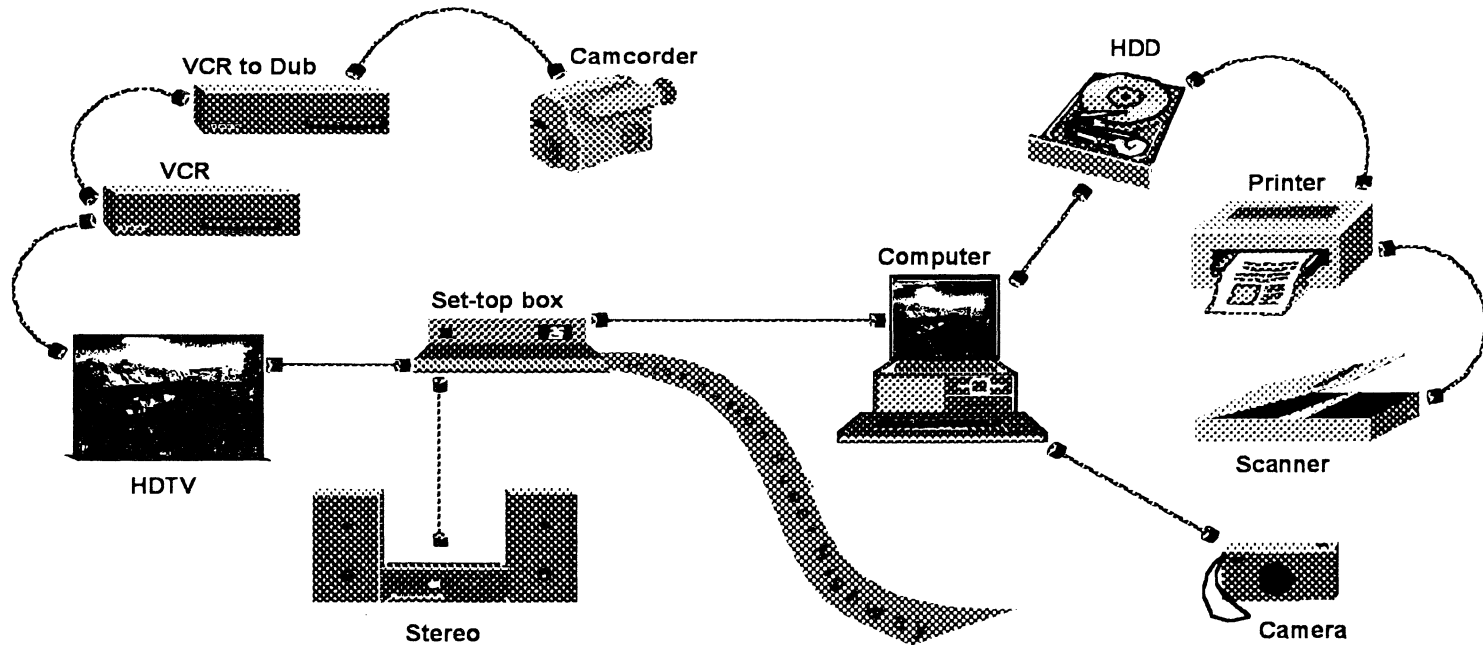
**1394 Market Development Manager**

**Texas Instruments**

# Isochronous Data Over PCI

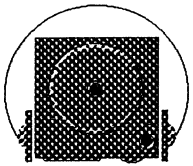
## Isochronous Data Providers

303

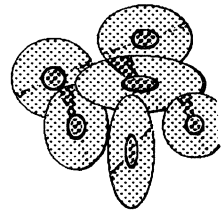
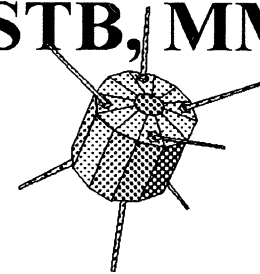
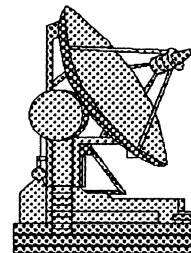
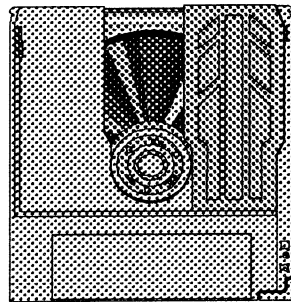
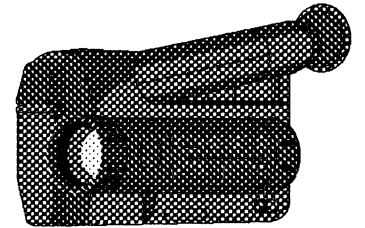


# Isochronous Data Formats

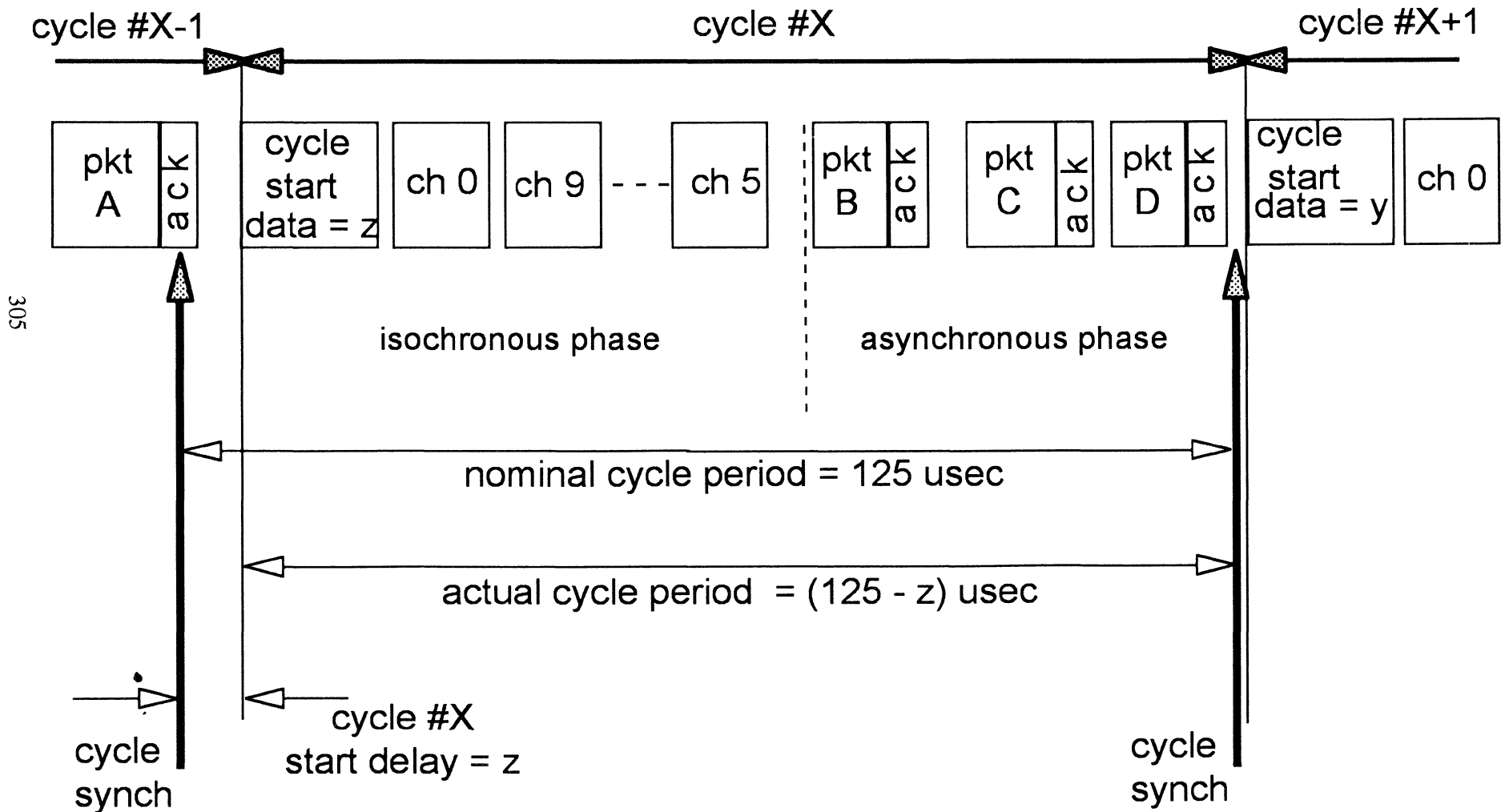
- ◆ **Raw data**
- ◆ **Audio: AC3, mLAN, others**
- ◆ **DV format, video (5:1) & audio**
  - **Digital Camcorders & VCRs**
- ◆ **MPEG-2**
  - **DVD, DVHS, DTV, DSS, STB, MMCD**



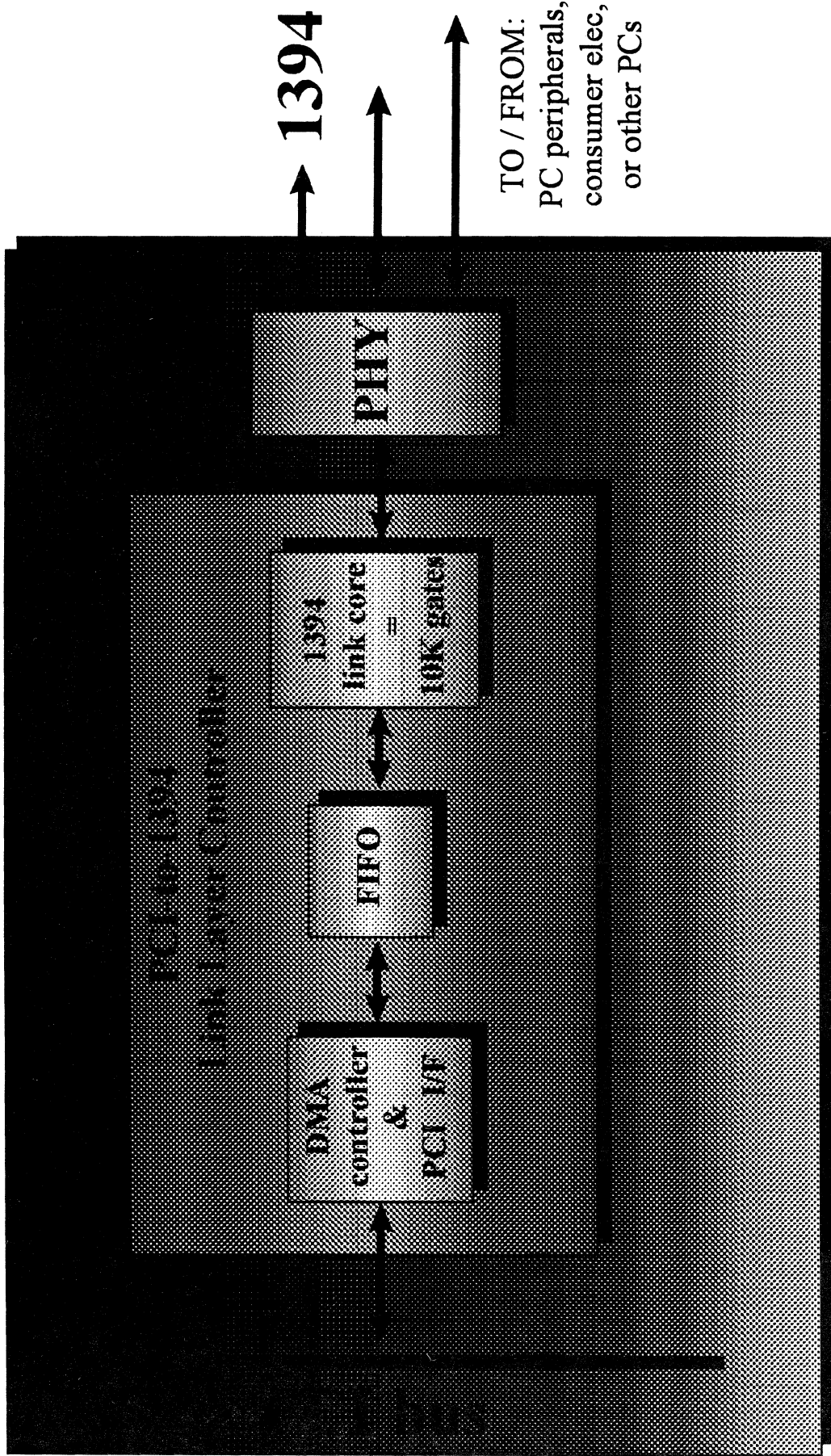
304



# 1394 Isochronous Cycle



# 1394 Data Flow in the PC

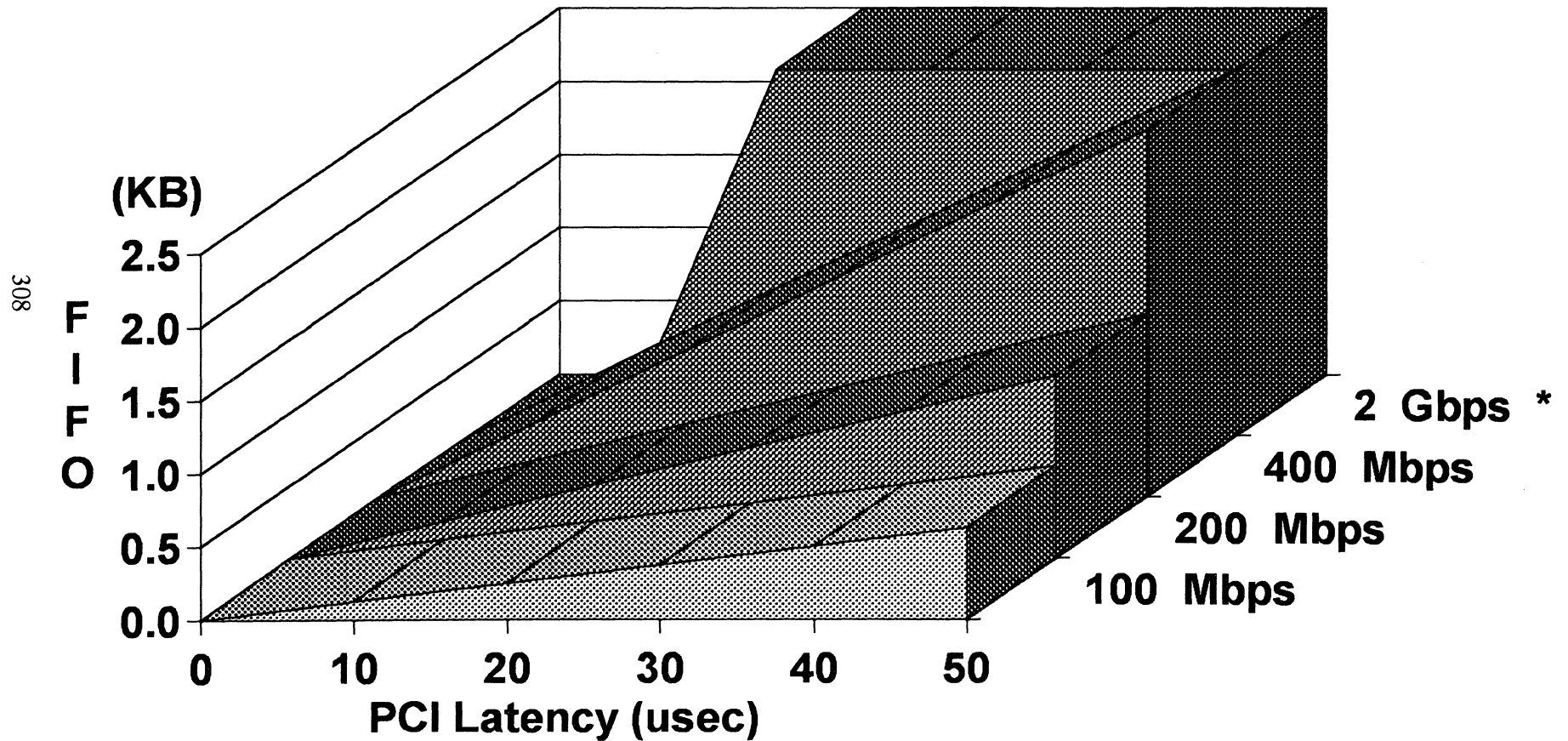


# PCI Latency Issues

- ◆ **PCI arbitration latency (arb to grant) and PCI Burst length**
  - **older PCI SIOs are very poor (~10MB/sec)**
  - **newer ones are much better (~100MB/sec)**
- ◆ **Number of PCI nodes active?**
- ◆ **PCI bus speed & width**



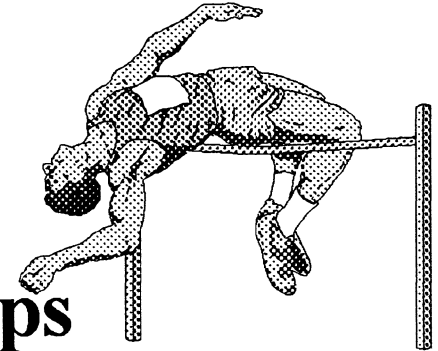
# PCI Latency vs. FIFO Size



\* 2 Gbps not recommended over PCI bus

# PCI and Other Solutions

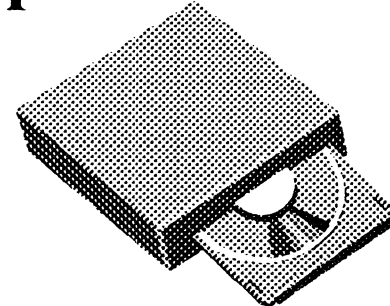
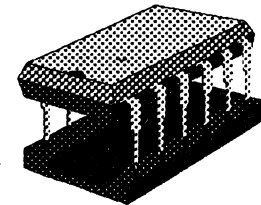
- ◆ **PCI's great for 100-400 Mbps**
- ◆ **Raising the bar: 1394 @ 2 Gbps**
- ◆ **PCI's future:**
  - **33 vs. 66 MHz & 32 vs. 64 bits wide**
- ◆ **Zoom Video port (ZV)**
- ◆ **Northbridge integration**





# Don't Miss THE Bus

- ◆ **1394 Consumer Electronics tidal wave in 1Q97, BE PREPARED!!!**
- ◆ **PCI-to-1394 chips are available today**
  - **100 & 200 Mbps now**
  - **400 Mbps & 2 Gbps very soon**
- ◆ **All of the pieces are in place**
  - **PCI-to-1394 Development kits**
  - **Silicon**
  - **Windows Software**



## **More info.**

- ◆ **<http://FireWire.org>**
- ◆ **<http://www.ti.com/sc/1394>**



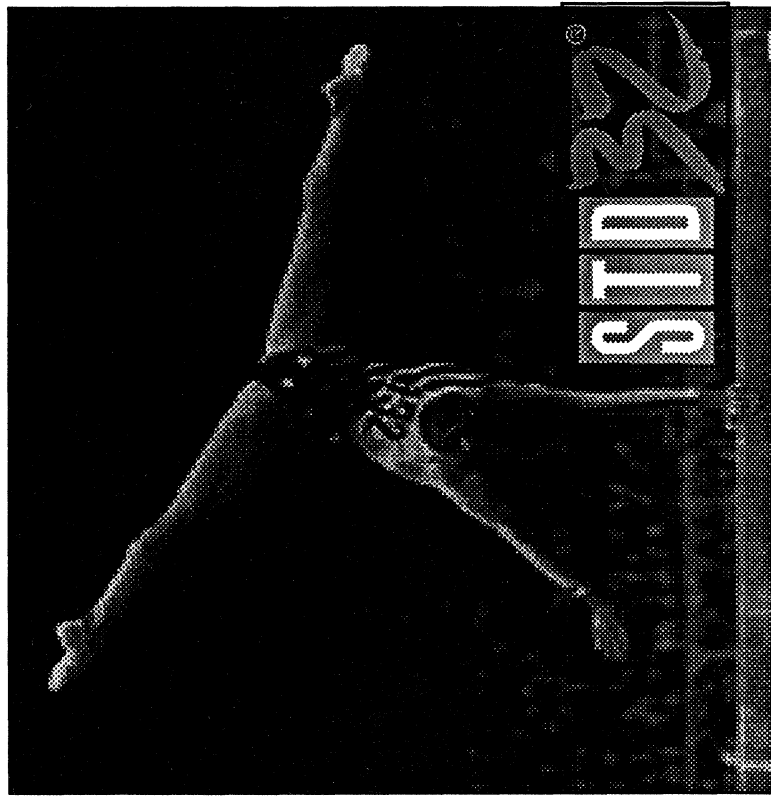
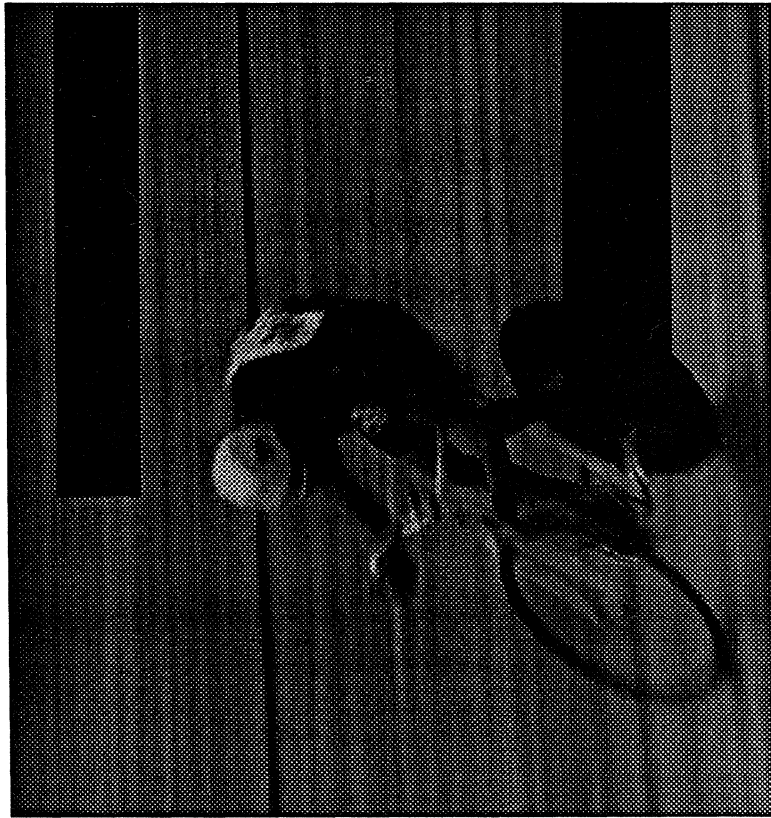
PCI - Spring 1996

# Overview

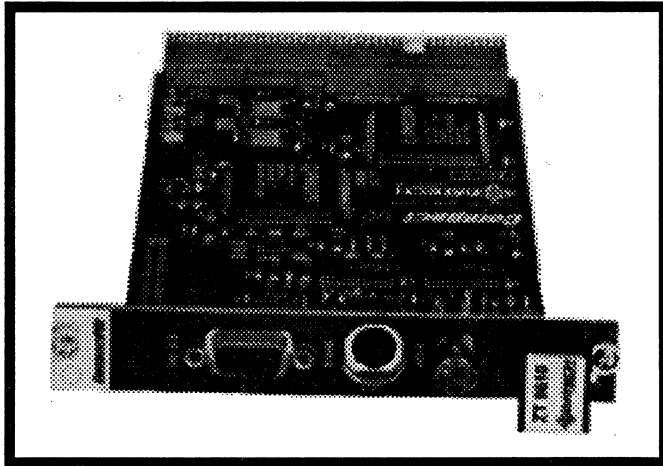
- ⇒ **Bringing PCI to the OEM embedded market**
- ⇒ **Creation of a new PCI Standard form factor**
  - ◆ **Birthing issues**
- ⇒ **Bridging to existing architectures**
  - ◆ **STD 32**



# Small, Rugged Computers for Control

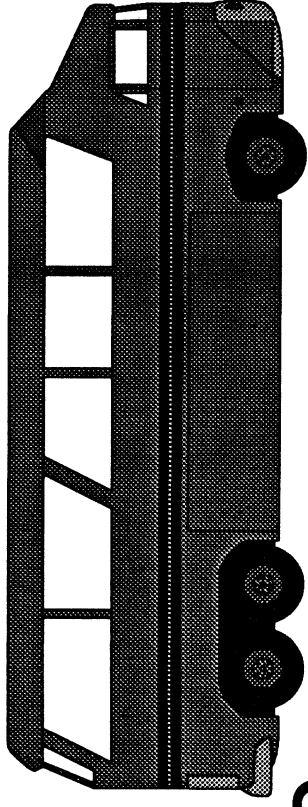


# What is CompactPCI?



- ⇒ CompactPCI = PCI + compact form factor
- ⇒ High-performance pin and socket connector
- ⇒ Industry standard Eurocard form factor

# What *CompaqPro* is Not



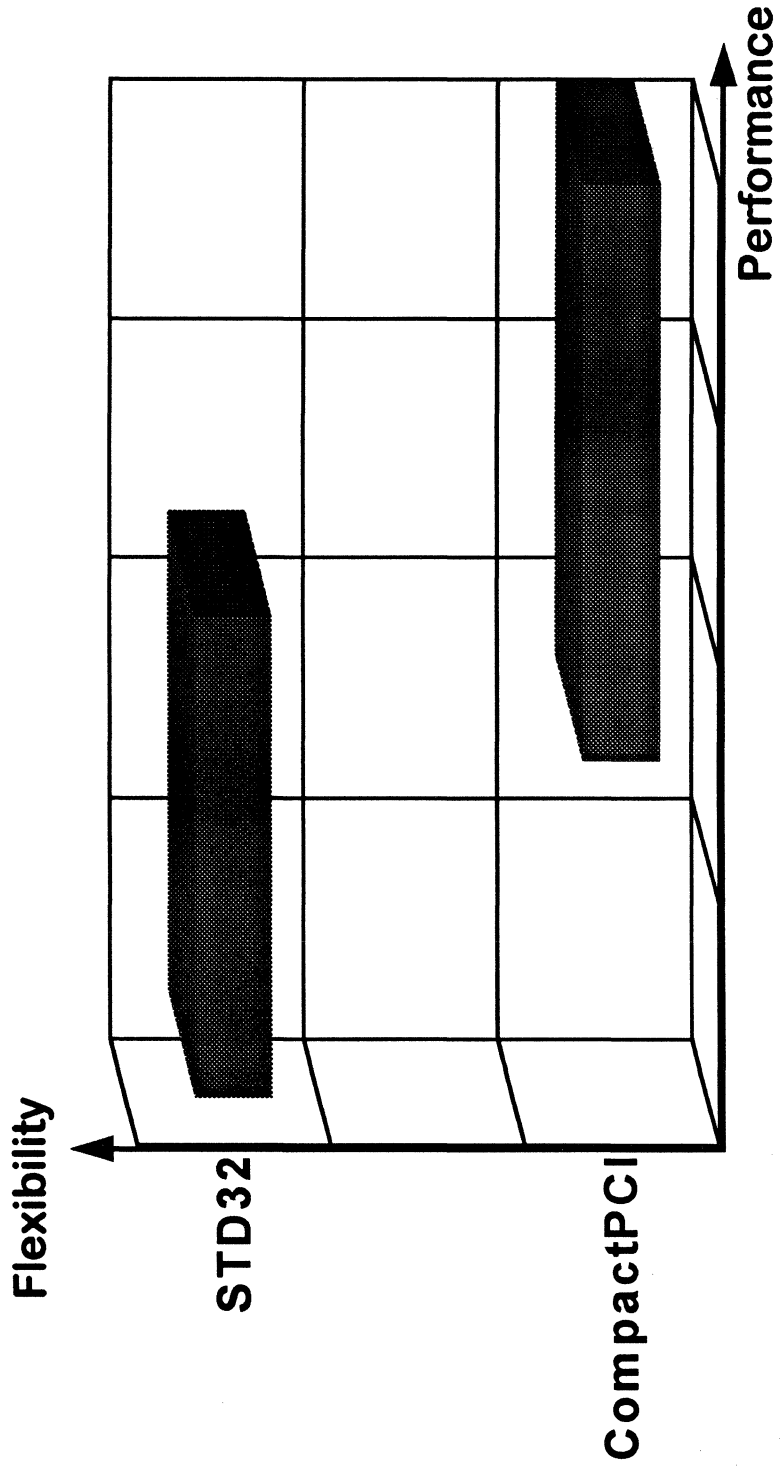
- ⇒ Large bus (slots)
- ⇒ Multiprocessing bus
- ⇒ For general purpose I/O
- ⇒ Easy to design custom adapters

# STD 32

- ⇒ Extension of the STD Bus
  - Well Established
- ⇒ PC software-compatible
- ⇒ Multiprocessing
- ⇒ Broad range of I/O cards
- ⇒ Simple To Design
- ⇒ Electrically similar to ISA/EISA



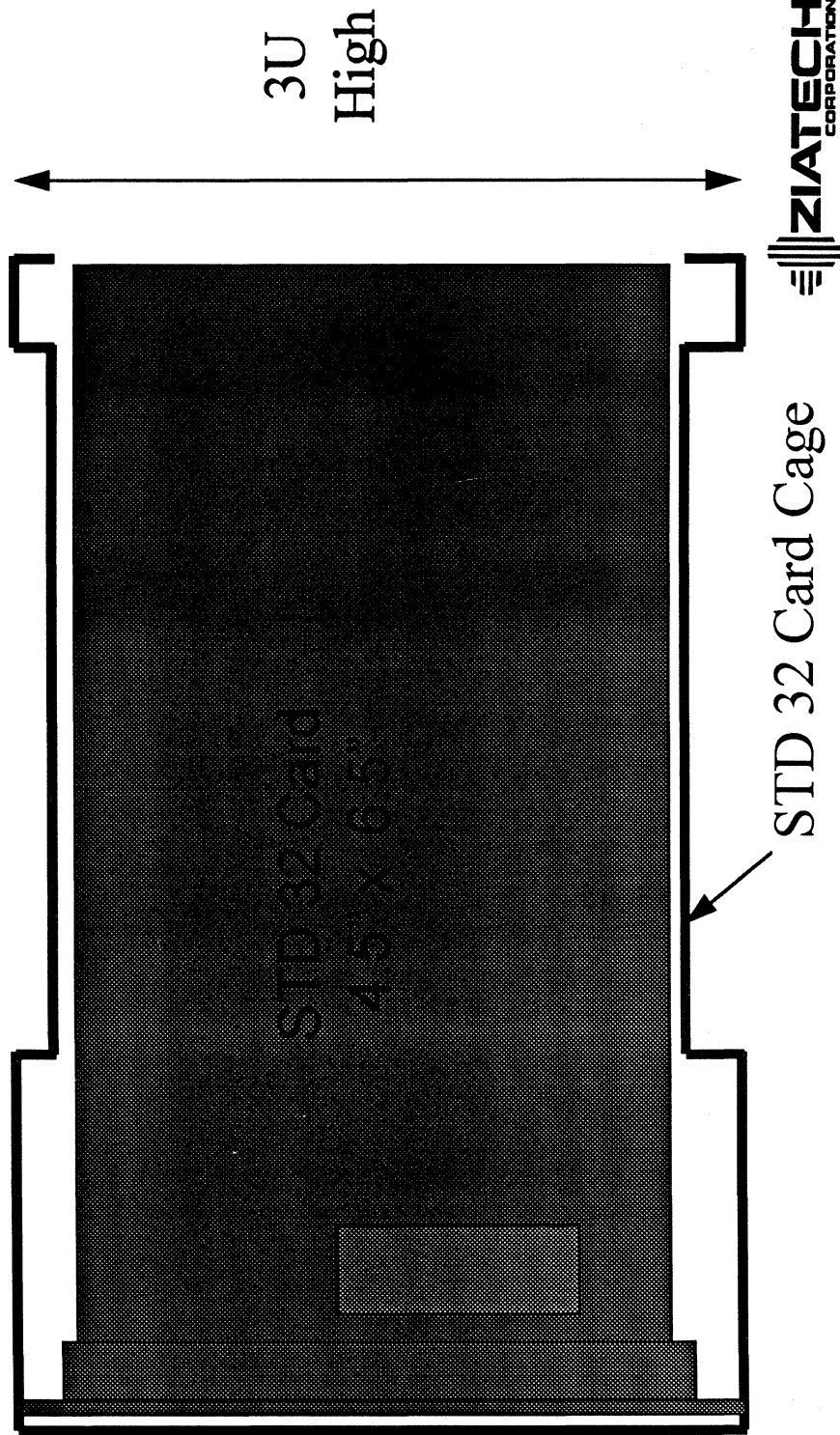
# CompactPCI and STD 32



# The need for a bridge

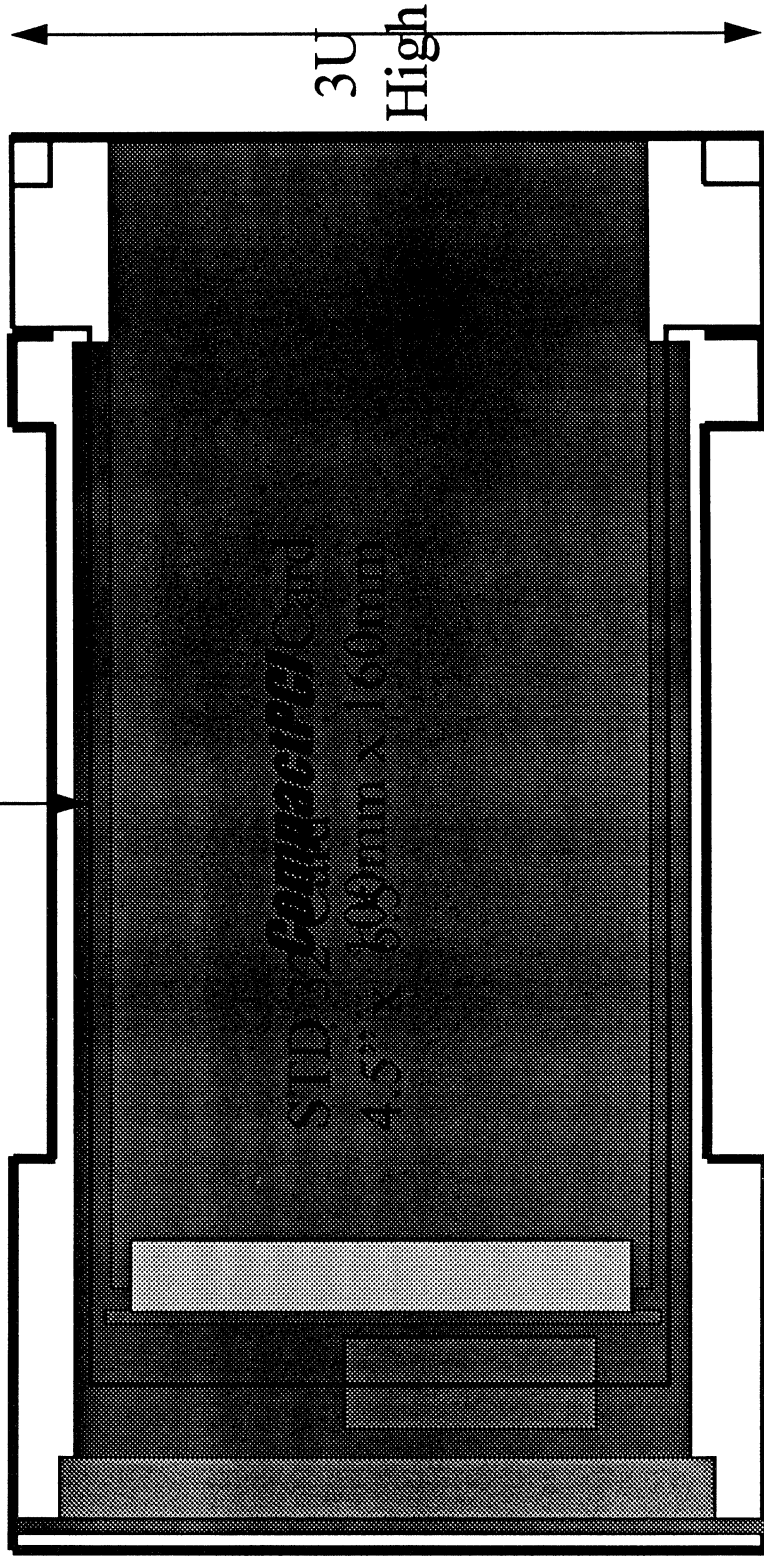
- ⇒ Combines the Bandwidth of CompactPCI with the Flexibility of STD 32
- ⇒ Keeps Fast and Slow I/O on Separate Buses
- ⇒ Allows for Multiprocessing
- ⇒ Growth Path for STD 32 Customers

# Mechanical Issues

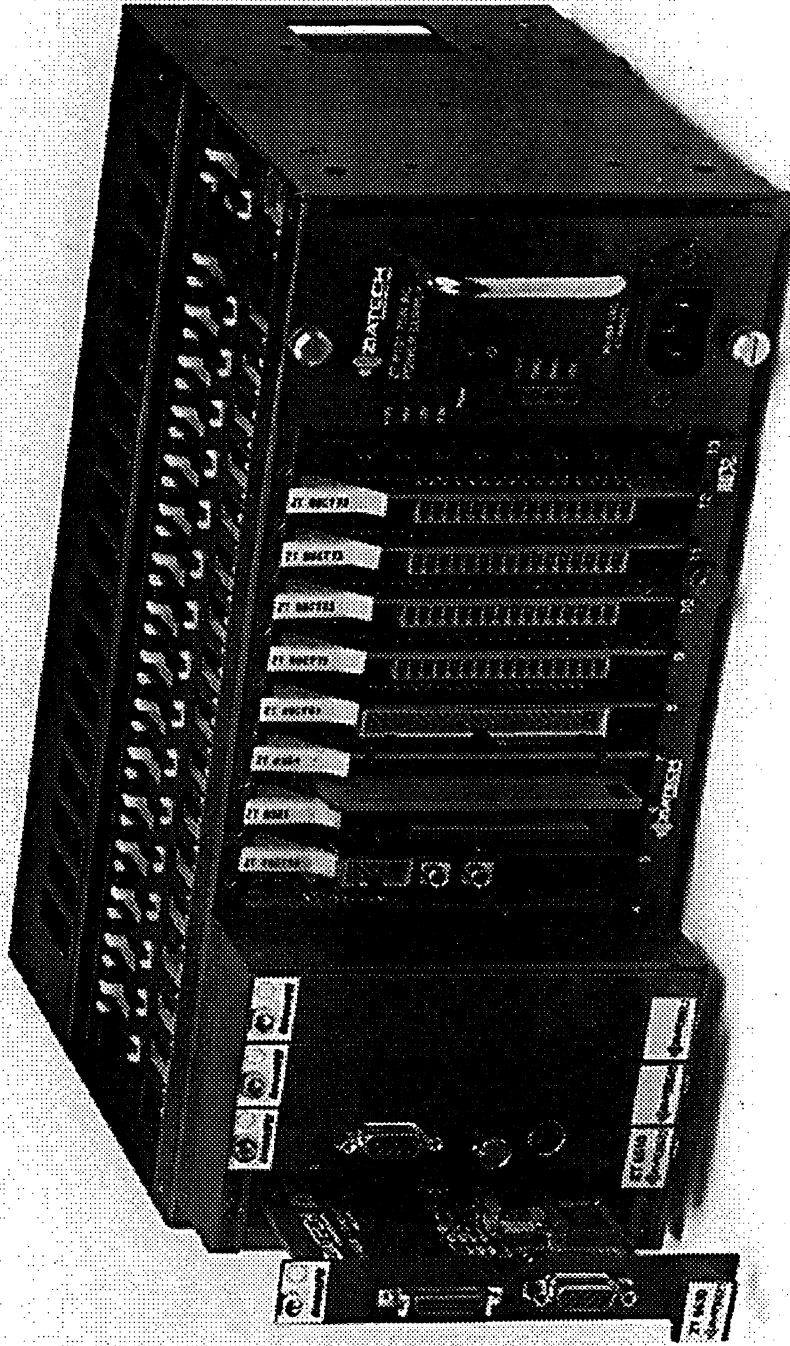


# Mechanical Issues

**CompactPCI** Subsystem

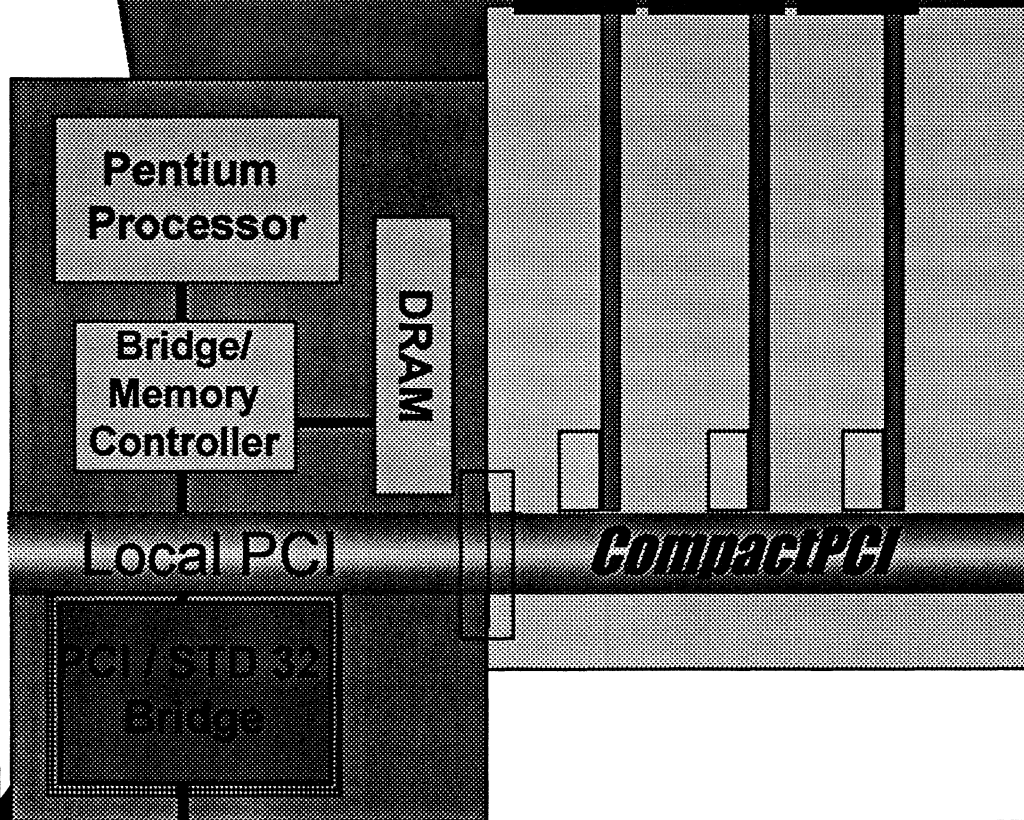


CompactPCI + STD 32



**ZIATECH**  
CORPORATION

# ZT 8906 Architecture



CompactPCI cards for  
Video, Network ,  
Vision etc..

323

Other STD 32 Processors, Digital and Analog IO, Serial  
MIL-1553, PLC Connections, Motion Control

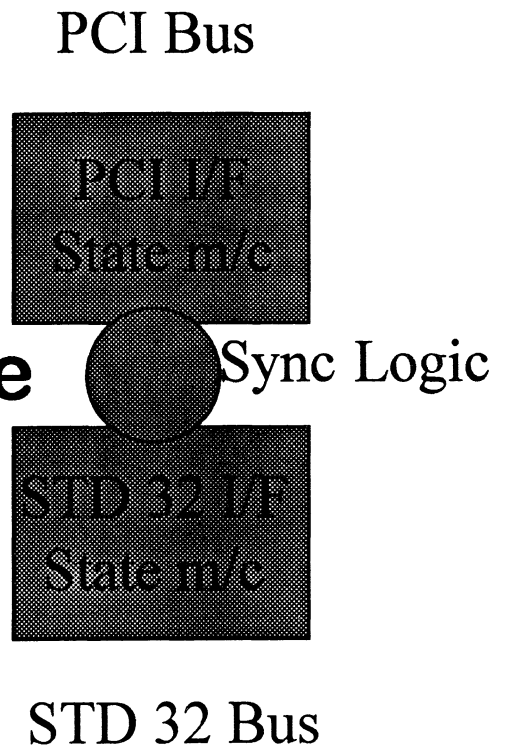


# Electrical Issues

- ⇒ Synchronizing two separate Synchronous buses
- ⇒ Providing PCI Plug & Play Configuration registers
- ⇒ Support for ISA peripherals on BOTH sides of the bridge
- ⇒ Legacy IO compatibility
- ⇒ Creating PCI parity cycles

# Electrical Issues (Cont.)

- ⇒ FPGA solution for ongoing flexibility
- ⇒ Separate State Machines
- ⇒ Coexisting with a subtractive bridge (ISA)





# Software

- ⇒ PCI Configuration BIOS
- ⇒ ISA and STD 32 peripherals
- ⇒ In-house BIOS resources eased implementation

# Summary

- ⇒ Bridging CompactPCI (for bandwidth) with STD 32 (for flexibility)
- ⇒ Mechanical AND Electrical issues
- ⇒ ... and software
- ⇒ Legacy issues are still important!

# Serial Storage Architecture

## A Low-Cost, High-Speed Serial Connection for Disk Subsystems

Edge Hawes  
Advisory Architect  
Storage Subsystems Development  
IBM Havant  
PO Box 6  
Havant, Hampshire, PO91SA  
United Kingdom  
  
+44 1705 486363  
Fax: +44 1705 499278  
  
adge@uk.ibm.com

February 1996

*This paper provides an overview of SSA (Serial Storage Architecture). Bringing a new and much-needed standard serial contender to the world of disk drive interfaces, SSA combines the speed and robustness of a comprehensive DC-balanced 8B/10B coding system with the low cost of CMOS driver electronics, the cabling benefits afforded by a low signal count, and the performance advantages obtained by use of a loop topology. The paper outlines the benefits and design details of the serial interface, and gives examples of its use in practice in large disk arrays. It also gives a status of SSA's standardization in ANSI, and an indication of the silicon and software that is available to SSA developers.*

---

## Introduction

### High Speed I/O Buses

As the performance of personal computers and workstations grows in leaps and bounds, so the requirements of their I/O bus increases dramatically. Since 1981, personal computer development has seen the standard I/O bus go from the humble 8-bit 'expansion bus' of the IBM PC/XT, through the ever-popular 16-bit 'Industry Standard Architecture' (ISA) of the IBM PC/AT, to the 32-bit EISA, Micro Channel, VESA Local-Bus and PCI buses. For the future, PCI defines a 64-bit extension, and VESA is also has a 64 bit version of its VLbus.

These developments recognize that as clock speeds of data transfer within a computer system reach reasonable economic maximums (where electromagnetic interference is to be minimized), the only way to increase the data transfer rate is to increase (and this usually means double) the data width.

### External I/O Connections

Going beyond the bounds of the computer's motherboard, however, it is a different story. Frequently, attachment devices such as CD-ROMs, tape drives and particularly extra disks and disk arrays need a fast, low-cost external attachment. While many 'out-of-box' connection methods exists, most notably the "Small Computer Systems Interface (SCSI)", speed improvement on such buses is not so easy. Increasing the data width, which in practice means doubling it, has two problems.

- The cable and connector almost double in size. Of necessity, such cables are invariably low-cost (ribbon cable or twisted-pair) and usually bulky anyway.
- The risk of crosstalk between data signals increases.

In addition, the ability to increase the clock speed on such buses is limited by the EMI and data skew complications, if an expensive shielded cable is to be avoided.

The SCSI specification has introduced versions which increase its clock speed and its data width allowing it to operate at up to 40MBytes/s, but it is recognized that further expansion possibilities are limited.

## The Serial Alternative

As the problem with expanding the effective data rate of such I/O buses are due to the physical size of the cable, an alternative approach is to *reduce* the size of the cable, ultimately to a single signal. As a narrow communication path usually involves serializing the wider data units that the computer uses, such connections are invariably referred to as **Serial** links or buses.

Given the performance advantages of the wide data bus, such a move would appear to be a backwards step. However, a serial I/O connection can have several advantages:

- The ASICs that implement the connection can have very few pins devoted to the connection itself. In most technologies it is the number of *pins*, rather than the number of *logic cells*, that dictate the ASIC cost.
- The ASICs have less of their circuitry devoted to the drivers and receivers of the signals. These components are often in short supply, are difficult to place on an LSI chip optimally, and usually consume a lot of power. Also, many such signals changing at once, as is characteristic of wide data buses, can cause severe 'ground bounce' problems.
- The connecting cable has less wire. Apart from the cost and bulkiness advantages, it means that more sophisticated shielding can be applied inexpensively than would be possible on multi-signal cables.
- With a single data signal there is no clock skew to consider as the data rate or the length of the cable increases.
- The I/O connection may be continued across other transmission media (than conventional copper wire), that don't lend themselves easily to parallel transfers (e.g. optical, radio, telephone).

These advantages can more than compensate for the reduced data width, and in turn allow more bandwidth expansion possibilities.

## How Many Signals?

It is tempting to think that a serial connection always consists of one wire. Unfortunately, the nature of the transmission medium, the speed of the data, and the requirements of the connection protocol make the 'one-wire-only' serial connection a rarity. The most famous serial link, RS232, requires some 9 connections! The bus frequently requires extra connections for the following reasons:

- A reliable transmission protocol often demands two-way, or *full-duplex* communication. The second communication path may be used for simple acknowledgement, for bi-directional concurrent data transfer, or both. When used in a loop, such as may be found in disk arrays, the second path may be used to double the bandwidth to that drive, or provides an alternative route to a drive whose connection has failed.
- The nature of copper wire means that high frequency signals can most effectively be transmitted using "differential pairs", wherein the same signal is transmitted with opposite polarity on the other wire of a pair. The two wires suffer similar noise and distortion corruption during transmission and the signal may be extracted by subtracting the output of one member of the pair from the other.
- Some serial links transmit the clock on a second signal. This means that reception of the data is simplified, and variation of the clock speed may be accomplished somewhat transparently (albeit with clock skew considerations). However, the addition of a second signal for a clock represents a 100% increase in signals required while adding little or no information content (in the classical sense).

Examples of serial buses include: Philips Inter-Integrated-Circuit bus (I<sup>2</sup>C), which is a 100Kbits/sec two-signal (clock and data) bus for connecting integrated circuits; IEEE P1394, an up to 393Mbits/s two signal (data and strobe) point-to-point connection for desktop computers; and Fibre Channel (FCS), a 1Gbits/s high-end fibre optic link for LANs (Local Area Networks) and WANs (Wide Area Networks).

---

## Serial Storage Architecture

Serial Storage Architecture (SSA) is a serial link designed especially for low-cost high-performance connection to disk drives and other peripherals. It is a two signal connection (transmit and receive), providing full duplex communication. It uses a self-clocking code, which means that the data clock is recovered from the data signal itself rather than being transmitted as a separate signal. For transmission along copper wire, it uses the differential pair method, requiring four wires, but it can also be transmitted along fibre optic cable.

The copper wire transmission rate is currently 20MBytes/sec in each direction, for a maximum of 25 metres cable length.

## SSA Components

Serial Storage Architecture currently comprises two components:

- SSA-PH/TL** The physical and transport layers. These are the physical and electrical specifications of the serial link and the low-level transport protocol, and is what is described in this article.

**SSA-SCSI** The SCSI-2 mapping. It is recognized that the *logical* aspects of the SCSI specification remain quite appropriate for addressing serially attached peripherals, and so this aspect of SCSI has been mapped to the physical SSA interface. This means that a transition may be achieved from SCSI to SSA with the absolute minimum of code rewrite.

In addition, it is planned that similar mappings will be provided for the SCSI-3 specification when it is approved by the American National Standards Institute (ANSI).

### **SSA Characteristics**

The SSA serial link has the following characteristics :

#### **Topology:**

A flexible addressing scheme that allows connections as strings, loops, and switched loops. "Hot swapping" is also permitted.

#### **Distance:**

Point-to-point connection for up to 25 metres (cable). Fibre-optic connections could support distances of up to 1km between nodes.

#### **Bandwidth:**

Full duplex communication. 20MBytes/s (200Mb/s) in each direction. The protocol allows for speed increases as technology permits. Work is currently under way to develop and standardize a 40MB/s version.

#### **Format:**

The unit of transmission is the frame, which can be up to 128 bytes. The minimum overhead per frame is 8 bytes, or about 6%.

#### **Reliability:**

The link is highly reliable. The design provides considerable detection of errors of all kinds, and a large amount of transparent error recovery.

#### **Physical:**

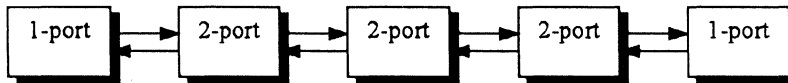
The cables and connectors have a small form-factor.

### **Topology**

The SSA design allows an extremely flexible assortment of connection options. SSA networks can be connected in simple strings or loops, or more complex switched strings or cyclic paths. This flexibility allows trade-offs to be made between cost, performance and availability.

This variation is afforded by three different types of SSA nodes:

- Single Port
- Dual Port
- Switch



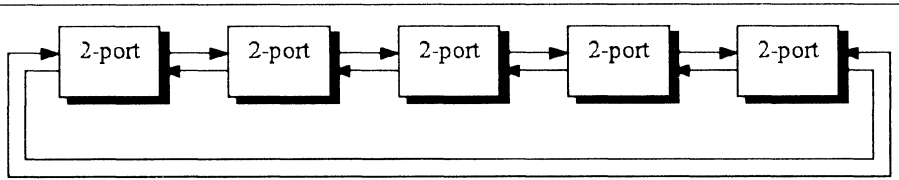
**Figure 1. An SSA String**

---

### **Strings**

A **string** is a simple linear network of two or more nodes, as shown in Figure 1. The port at either end can be single-port nodes, while the others are dual-port nodes.

A special case of a string is the *dedicated connection*, where two single-port nodes connect to each other across one link.

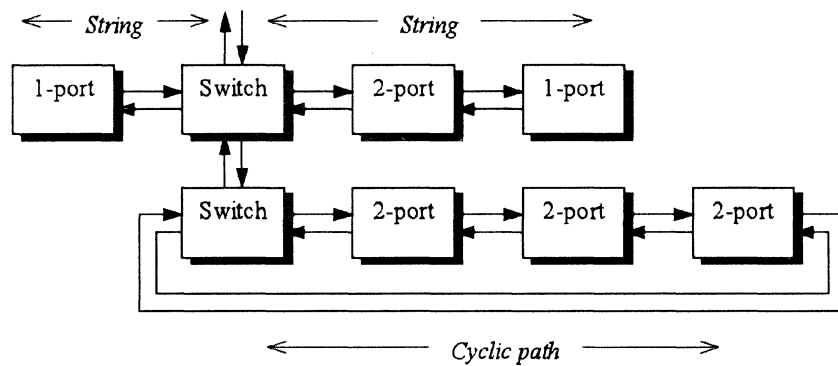


**Figure 2. An SSA Loop**

### Loops

The commonest form of connection is the loop, shown in Figure 2. A loop is a cyclic network containing only dual-port nodes. Loops have the benefit of higher bandwidth (there are two data paths available between any two nodes) and higher reliability (any single node can fail without prohibiting communication between nodes) than strings. In addition, a node may be inserted into the loop without breaking communication.

### Switches

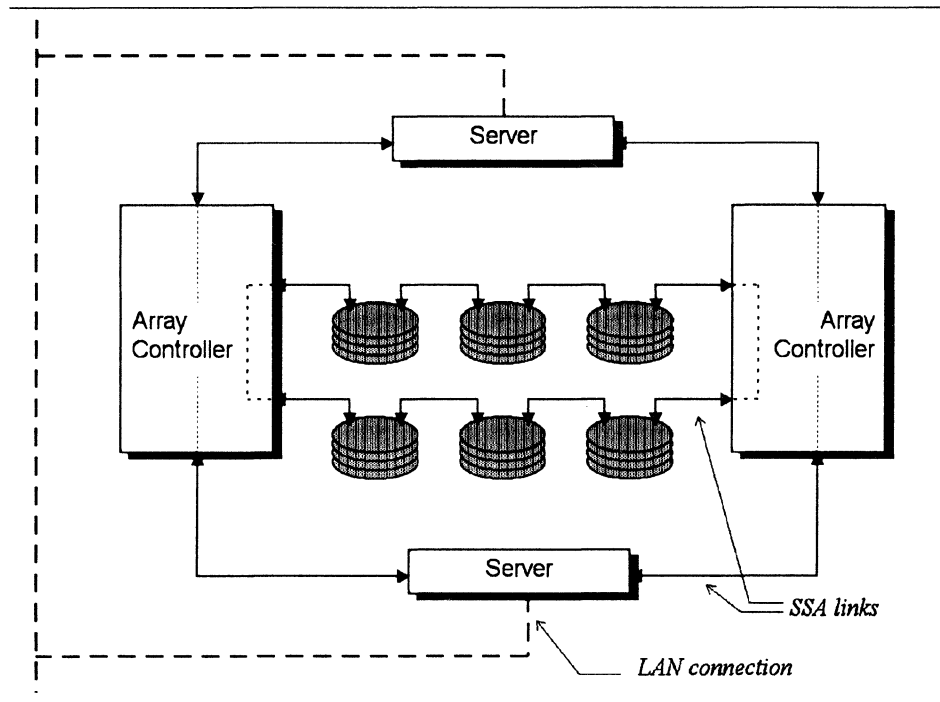


**Figure 3. An SSA Switched Network**

Figure 3 shows an example of a complex network involving a switch.

A switch can have up to 126 ports. Switches allow large numbers of nodes to be connected together, and also enable alternative paths to be established to provide fault tolerance.

Note that a switch network can also include other cycling paths; these are not loops by definition, as they involve other than dual-port nodes.



**Figure 4. A High Availability Server.**

### **High Availability Server**

As an example of an SSA network (usually referred to as a **Web**), Figure 4 shows a configuration for a **High Availability File Server**. Here, *high availability* means that the network is tolerant of a single fault; a failing link can be identified, and there may be some loss of bandwidth, but otherwise the operation of the network is not impaired.

This network contains two loops. The outer one connects the servers and the array controllers; the inner one connects the controllers and devices. *Spatial Reuse* (see “Spatial Reuse”) allows each path to provide up to 4 times the bandwidth of a single link.

### **Coding**

Despite the serial nature of the link, the unit of information transmitted remains the byte. Conventionally, an 8-bit is *serialized* and sent as a bit stream. However, if these bytes were serialized and sent one after another with no intervening control bits (as would be necessary for maximum throughput), the following problems would arise:

- There would be no way to recover and synchronize the data clock, especially with a stream of, say, all ‘0’s;
- There would be no indication of when a byte began and ended;
- Any errors occurring would be undetected and uncorrected;
- There would likely be a DC bias in the data signal, as more bits were sent of one polarity than the other.

To overcome these problems, a serialized bit stream is usually at a fixed clock rate, and is interspersed with control bits (e.g. Start, Stop and Parity) which go some way towards solving the first three problems at the expense of some bandwidth.

The last point, the DC bias, is usually ignored, and yet it is sometimes this one that prevents the serial link from moving to higher speeds. This is because as the data rate increases, the data signal contains more and more components at higher frequencies. To extract these signals reliably sometimes requires AC-coupled amplifier circuits at the receiver end. For example, the average value of DC-free signal (obtained by simple integration) can be used to provide the *slicing* point, the level that distinguishes a one from a zero. This is particularly useful for fibre-optic receivers, and is also the principle used in coding on Compact Disk and CD-ROMs.

## ***8B/10B code***

Because of these and other considerations, SSA uses a form of coding called '8B/10B' encoding. The name reflects the fact that the 8 bits to be coded become 10 bits of data. Thus there is a 25% redundancy in the signal, but as will be seen, this overhead is well spent.

Obviously, a 10 bit value can take on any of 1024 combinations, but many of those combinations are not valid, being excluded by the following rules:

1. There are no more than 5 consecutive bits of the same value (this applies between adjacent bytes as well as across a byte),
2. The maximum 'Digital Sum Variation' (**DSV**) is 6 (+3 to -3).

The run length limit ensures that the clock can be successfully recovered at all times, and will remain in synchronism.

The DSV is defined as follows: Counting a '1' as +1 and a '0' as -1, then a running count is kept as the bits are coded. The maximum value of this count minus the minimum value is the DSV. If this value is constrained, then the DC component of the transmitted signal is effectively zero; there are as many '1's as '0's in the stream overall.

For this to be achieved, the codes that are selected for transmission are dependent on the codes that have been previously been used, as well as the data that is to be transmitted. Thus each byte to be encoded can map to more than one code that is transmitted. This accounts for most of the redundancy in the 8B/10B coding method.

In practice, the 8-bit byte to be encoded maps to one or two codes. The byte is split into two parts, a 5-bit string and a 3-bit string. The substrings are then encoded using a 5B/6B code for the first substring and a 3B/4B code for the second. Each substring has either an exact number of '1's and '0's, or is unbalanced by 2. An imbalance is corrected by selecting the version of the next byte's code that has the opposite imbalance.

## ***Special Characters***

This coding method successfully encodes all 256 possible values for a byte, but also allows a further 12 codes to be included that obey the coding rules. Of these 12, three have the unique property of containing a string of bits within them ('0011111'b) that can never occur in any other bit position, either within a code or across a boundary between two code words. This property dubs them **Comma** codes, and may be used by the receiver to synchronize its clock to the correct codeword boundary. The remaining codes are used by the SSA protocol as 'special' characters, as they are readily identified as non-data.

The special characters in SSA are:

<b>FLAG</b>	Used to delimit the start and end of a <i>Frame</i> . It is also a comma character
<b>ABORT</b>	Prematurely aborts the transmission of a frame.
<b>SAT</b>	With SAT', used for arbitration.
<b>SAT'</b>	With SAT, used for arbitration.
<b>DIS</b>	Indicates the disabled state. It is also a comma character.
<b>ACK</b>	Acknowledges correct reception of a frame.
<b>RR</b>	Indicates readiness to accept another frame



# Using A Design Foundation for Flexible and Rapid PCI Interface Development

Leo K. Wong  
Applications Engineer  
Altera Corporation  
San Jose, CA  
Email: lwong@altera.com

## 1. ABSTRACT

*Developing a customized PCI design to one's exact specifications can be an expensive, if not daunting, venture, involving NRE costs, lengthy design and debug cycles, and months of ASIC turnaround time. Off-the-shelf PCI interface ASIC or PCI chipsets decrease the resources required for in-house development, but lack the flexibility for customization. That may in turn increase the components count on board, resulting in higher production cost and lower reliability.*

*To address these problems, a set of PCI interface designs has been created for use with complex programmable logic devices (CPLD). These reference designs serve as foundations for PCI interface applications, asserting the necessary state machine controls and bus transaction signals. Complex Programmable Logic Devices (CPLD), coupled with this customizable PCI design foundation, provide a highly integrated methodology that can be tailored to the end application needs with minimum development effort and time.*

*This article describes the major features of this set of designs, and also covers important design considerations in developing a CPLD based PCI application. These considerations include device performance, resource utilization, electrical characteristics and other desirable features such as open drain I/O and on-board RAM. Finally, the paper then describes the*

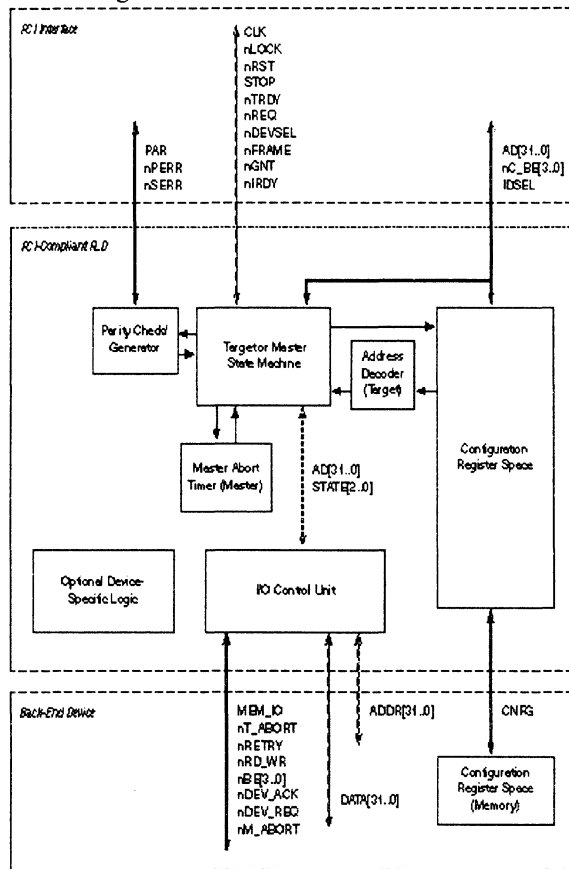
*strategies this set of reference designs employ to address these issues and how one might customize the function to his/her particular needs.*

## 2. DESCRIPTION OF THE DESIGN FOUNDATIONS

The PCI Design Foundations contains high-level behavioral descriptions files of both Target and Master. Under each type of interface are the design files optimized for different CPLD architectures. These design files, written in AHDL, take advantages of the unique features associated with each device family. These special features includes parallel and shared expanders for MAX 7000E; carry and cascade chains for FLEX 8000A; and the identity comparator for FLASHLogic. Associated with each design are the Assignments and Configuration Files (ACF) developed for each family. The ACF files direct MAX+PLUS II, the Altera PLD development system, to synthesize and place logic intelligently, enabling the underlying components to be PCI compliant.

In addition, sample test vectors, written in the form of waveforms, are provided as foundations of extensive functional and timing verifications. The verification can be performed in MAX+PLUS II or other standard EDA tools.

Figure 1. PCI Reference design Architecture Block Diagram



## 2.1 Target Interface

The PCI Target interface is a passive data path between the PCI bus and the back-end application. As a passive interface, it can not initiate a data transfer.

In accordance with PCI Spec. 2.1, the Target reference design respond to all valid transactions as indicated by the  $C\_BE[3..0]$  signals. In addition, the Target reference design supports a 32 bit Data Bus with 128 MB of address space. The address space is determined by 27 bits of address which compares to the 27-bit Base Address Registers (BAR) of the configuration space. The BAR determines the address of the

back-end application. It can be changed by executing a Write\_BAR operation, enabling auto configuration, alleviating the need for DIP switches.

## 2.2 Master Interface

The PCI Master interface is an active data path. It can initiate data transfers to and from Target interfaces. When the back-end application requests a transaction, the Master responds by requesting the system arbiter for bus control. After the arbiter grants control, the Master provides the necessary interface Control signals, address, and Command/Byte Enable to the PCI bus. Since it issues addresses, the Master reference design is not required to provide address decoding. It supports the same types of transactions that the Target reference design supports.

The Master reference design supports the 32-bit bus data transfers and a 6-bit addressing for the Master configuration space. The 6-bit configuration address is sufficient for addressing 64 DWORDs which maps to the 256 Byte configuration space.

The reference designs are easy to modify, allowing designers to tailor to their system specifications. For instance, PCI protocol supports three decoding modes: fast, medium, and slow. This set of design foundations implement medium decode, where the address is decoded on the next clock cycle after  $FRAME\#$  is asserted. PCI designers might want to change it to a different decoding mode. Another example is the address space. If the address space is smaller, the number of address bits and the corresponding BAR can be reduced.

### 3. DESIGN CONSIDERATIONS

#### 3.1 Timing Considerations

Strict timing requirements are imposed on PCI bus signals. In particular, the Specification 2.1 requires that PCI signals meet 7 nsec external Setup time, 0 nsec external Hold time, and 11 nsec Clock-to-Out time. Furthermore, it requires 33 Mhz operation for off-the-shelf PCI cards. Due to these timing restrictions, only high performance components with predictable timing should be considered for PCI interface applications. High density programmable logic devices such as the MAX 7000E-10P, FLEX 8000A-2 devices, and the EPX 8160-10 meet these requirements.

These PCI timing requirements do not apply to the signals that go to or come from the back-end application. It is only the signals on the PCI bus that need to meet the specification. The only constraint that the back-end has to meet is the system frequency of operation. For off-the-shelf cards, this requirement is 33 Mhz. Proprietary or closed PCI systems could be designed for a lower frequency of operation.

The PCI reference designs implement several design strategies to meet PCI Specifications with the target devices. These strategies are employed for designers in the form of Assignment and Configuration File (ACF). Explanations for those assignments are given by the following sections.

##### 3.1.1 Target Timing

For MAX 7000E, the -10P speed grade devices easily meet the required 0 nsec external Hold time and 11nsec Clock-to-Out when the tri-state is enabled one

cycle before the Address/Data is clocked out. The way MAX 7000E family meet 7 nsec Setup time requirements depends on the decoding mode.

In medium decoding mode, the address get decoded on the next clock cycle with reference to FRAME#. If the address needs to be decoded on the same clock cycle as the FRAME#, the Address/Data bus lines and Command/Byte Enable lines are registered before it gets decoded. The interface control signals are not registered to allow the interface to respond on the same clock cycle. The Setup time can then be met by using parallel expanders on the state machine decode. Parallel expanders are product term routed from an adjacent Logic Cell, as opposed to shared expanders, which are product term routed from the same Logic Array Block.<sup>1</sup>

In the FLEX 8000A family, the basic building block of the devices are called Logic Elements. These Logic Elements are interconnected together by continuous rows and columns channels. At the end of rows and columns are I/O elements, each contains an I/O cell register that feed an I/O pin. The delay from column pins to Logic Elements and the delay from row pins to Logic Elements are fixed but different. In order to meet the 0 nsec external Hold time in FLEX devices, PCI bus signals are assigned to column pins to generate a greater delay difference between clock-to-register and data-to-register signals. For the 11 nsec of Clock-to-Out time requirement, outgoing signals to the PCI bus should be latched from I/O cell registers. Furthermore, tri-state is enabled one cycle before the data is clocked.

To meet the 7 nsec Setup time, the Address/Data and Command/Byte Enable are registered before they get decoded.

---

<sup>1</sup> See detail descriptions of MAX 7000E device architecture in Altera 1996 Databook.

Since FLEX 8000A is a register-rich family, one-hot state encoding scheme is used for the target state machine. Furthermore, the reference design take advantage of the cascade chain, a dedicated path between two Logic Element, to speed up control signals decoding.<sup>2</sup>

### 3.1.2 Master Timing

With MAX 7000E, the same strategies as used in the Target apply. The only difference is the absence of address comparison to the BAR, resulting in less register usage.

In the case of FLEX 8000A, since Master interface has a more complex state machine decode than the Target. In order to meet the 7 nsec Setup, interface control signals are registered, resulting in one clock cycle latency. The other strategies for the Target also apply.

### 3.2 Logic Usage Considerations

In addition to timing specifications, PCI bus interface requires certain number of logic capacity and I/Os. Table 1 shows the logic cell and I/O utilization for timing optimized Target and Master reference designs in some sample programmable logic devices.

**Table I:** PCI Utilization on Different Devices

PCI Interface	Device	LCELL Utilization	No. of I/Os
Target	EPM7160E-10P	155 (96%)	98
Target	EPF81188A-2	306 (30%)	101
Master	EPM7160E-10P	123 (76%)	103
Master	EPF81188A-2	285 (28%)	134

Note that in the higher density devices, plenty of logic capacity is available for additional custom integration.

<sup>2</sup> See detail descriptions of FLEX 8000A device architecture in Altera 1996 Databook.

### 3.3 Other Resource Considerations

In addition to I/O pins and logic cell usage, PCI interface logic necessitates a certain number of control signals, including clear, preset, and output enable. Designers need to pay particular attention to the output enable signals needed to control the data flow. It is important to allocate the OEs required to the system specification. The Target and the Master don't have the same output enable requirement. These are itemized in the following subsections.

#### 3.3.1 Target

Table II and Table III shows the required output enable and their functions as defined by PCI specification Rev 2.1 .

**Table II.** Signals that Target handles:

Signals	Equations
OE [AD[31..0]]	(S_data + Backoff) & Tar_dly *(cmd = read)
OE[TRDY#]	Backoff + S_data + Turn_ar
OE[STOP#]	Backoff + S_data + Turn_ar
OE[DEVSEL#]	Backoff + S_data + Turn_ar
OE[PAR]	OE[AD[31..0]] (delayed by 1 clk)
OE[PERR#]	R_perr + R_perr (delayed by 1 clk )

Output enable signals can be consolidated if that are driven by the same logic. Resources are conserved as follows:

1. AD\_oe => controls AD[] and PAR
2. DATA\_oe => controls the DATA[]<sup>3</sup>
3. PERR\_oe => controls the PERR#<sup>4</sup>

<sup>3</sup> DATA[] represents 32 bi-directional pins interfacing with the back-end function.

<sup>4</sup> PERR# and SERR# are not required (PCI Local Bus Specification Rev 2.1) for the following classes of devices: Devices that are designed exclusively for use on the motherboard or planar; e.g. chip sets. System vendors have control over the use of these devices since they will never appear on add-in boards.

Devices that never deal with or contain or access any data which represent permanent or residual system or application state, e.g. , human interface and video/audio .devices. These devices only touch data which is a temporary representation

4. TRDY\_oe => controls the TRDY#, DEVSEL#, and STOP#

### 3.3.2 Master

**Table III.** Output enables required in a Master Interface:

Signals	Equations
OE[FRAME#]	ADDR + M_data
OE[AD[31..0]]	(S_data + Backoff) & Tar_dly *(cmd = read) if ADDR drive address if M_data drive data if DR_bus if (step * request ) drive address else drive lines to a valid state
OE[IRDY#]	(previous) M_data + ADDR
OE[LOCK#]	Own_lock & M_data + OE[LOCK#] * (FRAME# + !LOCK#]
OE[C_BE[3..0]]	ADDR + M_data + DR_bus if ADDR drive command if M_data drive byte enable if DR_Bus if (step * Request ) drive address else drive lines to a known state
OE[PAR]	OE[AD[31..0]] (delayed by 1 clk)
OE[PERR#]	R_perr + R_perr (delayed by 1 clk )

The output enables for Master can be grouped as follow:

1. FRAME\_oe => controls FRAME# and C\_BE[]
2. IRDY\_oe => controls IRDY#
3. AD\_oe => controls AD[] and PAR
4. DATA\_oe => controls DATA[]
5. PERR\_oe => controls PERR#
6. SERR\_oe => controls SERR#

### 3.3.3 Master & Target

The following Output Enables are required for a combined Master and Target interface.

1. FRAME\_oe => controls FRAME# and C\_BE[]

---

(e.g. pixels) of permanent or residual system or application state, and therefore, are not prone to create system integrity problems in the event of undetected failure.

2. IRDY\_oe => controls IRDY#
3. TRDY\_oe => controls the TRDY#, DEVSEL#, STOP#
4. AD\_oe => controls AD[] and PAR
5. DATA\_oe => controls DATA[]
6. PERR\_oe => controls PERR#
7. SERR\_oe => controls SERR#

Combined Master and Target interface can be implemented in a single EPM7160EQC160-10P if PERR# and SERR# reporting is not required. Otherwise, an additional EPM7032-7 can be used for the implementation. For FLEX 8000A, an additional EPM7032-7 can be used to complement the output enable buffer.

Note that the back-end bi-directional DATA[] bus can be divided into two 32-bit input and 32-bit output bus. This scheme will increase the number of I/O pins used but reduce the number of output enable required.

## 4. MULTI-CHIP SOLUTION

A single chip implementation of a PCI interface is preferable, but a multi-chip implementation is feasible. There are many reasons that make a multi-chip implementation necessary. They are insufficient logic capacity, insufficient pin count, insufficient output enable, or a combination of the above.

There are Loading, Timing, and Layout issues to deal with on a multi-chip PCI interface design. These topics are discussed in the following subsections.

### 4.1 Multichip-Loading

The PCI Local Bus Specification limits the load on a PCI signal to a single pin. Therefore, when design is partitioned,

none of the PCI interface signals can drive multiple pins.

One aspect of the multi-chip scheme that cannot be avoided is multiple clock fanout. Using one of the many available clock distribution chips on the market limits the load on the PCI clock to the specified 12pF while providing multiple clocks.<sup>5</sup>

#### 4.2 Multichip-Timing

Irrespective of the implementation, the PCI interface must meet the specified Setup ( $T_{SU}$ ), Hold ( $T_H$ ), Clock-to-out ( $T_{CO}$ ), and Register Frequency for off-the-shelf card. These requirements dictate how the PCI design is partitioned in a multi-chip implementation. The control signals generated from one chip must be available to the other chip in time.

First, the Output Enable signals generated from one chip has to be able to enable a tri-state buffer on another chip in less than (11 nsec - Output driver delay). For example, an Output Enable control signal from an EPM7032-7 will take 7 nsec to get to the tri-state OE control of EPM7128E-10P. This number was calculated by adding:  $T_{CO}$  of EPM7032-7 + ( $T_{i/O}$  +  $T_{pia}$  +  $T_{ioe}$ ) of EPM7128E-10P. Notice that this is less than 9 nsec (11 nsec  $T_{CO}$  - 2 nsec  $T_{od1}$ ). A similar calculation can be done between FLEX 8000A devices, and it will show that this requirement is also satisfied. FLEX 8000A meets this constraint irrespective of whether the outgoing Output Enable signal is from an I/O cell register or not. A combination MAX 7000E and FLEX 8000A is also feasible.

---

<sup>5</sup> Vendors such as Motorola and Cypress make devices for this purpose. Examples are CY7B991/2 Programmable Skew Clock Buffers from Cypress Semiconductor, or MC88915\*70 Low Skew CMOS PLL Clock driver from Motorola.

Secondly, signals from one chip that are needed for implementing a decode in another chip needs to be available within the required 30 nsec period.  $T_{CO}$  for MAX 7000E-10P is 5 nsec, and that of MAX7000-7 is 4.5 nsec. FLEX 8000A devices have very fast  $T_{CO}$  from I/O cell register, and comparable  $T_{CO}$  from a core register to a column pin. Refer to the FLEX8000 Data Sheet for the Timing Model and timing values. The chip implementing the decode has 30 nsec -  $T_{CO}$  time to decode the logic and meet the internal register setup.

#### 4.3 Multichip-Layout

PCI Local Bus Specification 2.1 requires that signal trace to a PCI interface chip be no longer than 1.5 inches for a 32-bit implementation. The PCI Connector Pinout is given in Table 4-11 of the Rev 2.1 specification. In order to meet the 1.5 inches requirement with this pinout, PCI chips have to be mounted on both sides of the board.

### 5. Burst Mode Operation

To realize the full performance potential of the PCI bus, zero wait state burst mode operation must be incorporated in the design. High speed CPLD such as EPF81188A-2 can execute zero wait state read *and* write operations for an indefinite period of time, thus supporting the maximum transfer rate of 133 MB/s. The burst design for other Altera devices will be offered in second half of 1996.

#### 5.1 Burst Mode Signals

This target interface performs a basic PCI burst mode protocol. There are three handshake signals between the target interface chip and the back-end: `wr_wait`, `rd_wait` and `i_wait`. They are defined in Table IV.

**Table IV** Burst Mode Back-end control signals

wr_wait	Input signal to our chip from the back-end indicating that the back-end can not <b>accept</b> data this clock cycle. It is an active high signal.
Rd_wait	Input signal to our chip from the back-end indicating that the back-end can not <b>provide</b> data this clock cycle. It is an active high signal.
i_wait	Output signal from our chip to the back-end indicating that the Master can not accept or provide data this clock cycle. It is an active high signal.

Burst addressing is handled by the back-end application. The target interface provides the first address. Currently, the PCI burst interface does not have data FIFO or register bank implemented. In register-intensive devices such as FLEX 8000A, these functions can be added as required.

### 5.2 Error Recovery in Burst Mode

The current burst mode implementation does not limit length of data transfer. At any intermediate point in the data transfer, a parity error can be detected. PCI specification does not explicitly state what should be done in this case. There are, however, two scenarios that can be explored. The choice is up to the hardware designer and BIOS developer:

The first option is to perform a complete retransmit. The advantage of this approach is that designers don't need a counter to keep track of the address. The second option is to restart transmission where the error was detected. It requires a loadable counter that is wide enough for the address space. The starting address of the data transfer gets loaded to the counter, and it keeps track of the address of succeeding data transfers.

The Target burst reference designs implement the first recovery scheme. When a burst is interrupted, the transaction

terminates and it starts from the very beginning when the burst restarts.

### 5.3 Implementation

This section describes the handshaking between the PCI bus and the back-end during burst write, burst read and master initiated termination.

#### 5.3.1 Burst write transaction:

During a burst write, the target interface outputs new data on the DATA bus every clock cycle to the back-end when IRDY# is asserted unless the back-end requests a wait, in which case, an input signal wr\_wait from the back-end will be received. The target interface de-asserts TRDY# one cycle after wr\_wait is observed on the rising edge of clk. Data on the Data bus is one cycle delayed version of the data on the AD bus. Figure 2 is the timing diagram for this transaction.

When IRDY# is de-asserted, the target interface generates an i\_wait signal to the back-end the same cycle when IRDY# is observed. In this case, the back-end will hold the current address when i\_wait is observed on the rising edge of clk. Figure 3 is the timing diagram for this transaction.

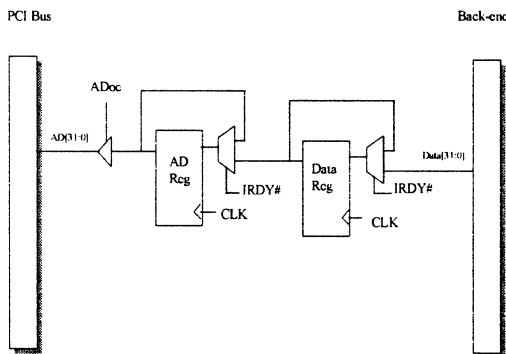
#### 5.3.2 Burst read transaction:

During a burst read, the target interface receives new data on the Data bus every clock cycle when IRDY# is asserted unless the back-end requests a wait, in which case, a rd\_wait will be received from the back-end. The target interface de-asserts TRDY# one cycle after rd\_wait is observed. This will cause the PCI bus to enter a wait cycle. The back-end holds its data on the Data bus after asserting rd\_wait. Figure 4

shows the timing diagram for this transaction.

When IRDY# is de-asserted, the target interface generates an i\_wait signal to the back-end in the same cycle when IRDY# is observed. In this case, back-end will hold the data when i\_wait is observed on the rising edge of clk. Data on the AD bus is NOT a one cycle delayed version of the data on the Data bus. In order to provide correct data to the AD bus, both the Data register and the AD register must output the previous data when IRDY# is de-asserted. Figure 5 shows the timing diagram for this transaction. Figure 6 is a block diagram of the data path for this operation.

Figure 2. Datapath for burst read



### 5.3.3 Master initiated termination:

When FRAME# is de-asserted and IRDY# is asserted (indicating the last data transition), the target must transfer this last data when it is ready. After this transfer, both IRDY# and TRDY# are de-asserted, and the PCI bus goes into the turn around state. The PCI target control signals TRDY#, STOP# and DEVSEL# are all de-asserted during the turn around cycle and are tri-stated the following cycle.

## 6. OTHERS

### 6.1 Implementing Open-Drain (O/D) pin:

Section 2.1 of the PCI Local Specification 2.1 states that certain pins on the PCI bus must be open-drain. This allows several devices to share a signal as a wire-OR. These signals consist of SERR# and INTA# - INTD#. Note again that SERR# and INTA# - INTD# are not necessarily required. INTA# - INTD# signals are used by PCI devices to assert an active-low signal to the system interrupt controller. The system designer determines whether this is required or not.

EPX8160 supports open-drain outputs. MAX 7000 and FLEX 8000 devices implement the open-drain structure by using an output enable to control an output pin that is configured to drive low. In effect, when SERR# needs to be asserted, the output is enabled. A pull-up resistor on the board ensures that the SERR# signal on the PCI bus is at a logic high in its quiescent state.

### 6.2 Low Power Options

MAX 7000 and FLEX 8000A PCI devices can be used with 3.3 V I/O option. This only impacts the Output Driver delay time, increasing it by 0.5 nsec. The Setup and Hold time remain unchanged when the 3.3V I/O option is used.

## 7. CONCLUSION

As the provider of the highest density and highest performance programmable logic devices, Altera has been proven to bring the benefits of time-to-market and flexibility to designers. With the introduction of the PCI reference design,



unprecedented level of design re-use and customization power are brought to the PCI design community.

#### ***AUTHOR BIOGRAPHY***

Leo Wong is an applications engineer with Altera Corporation, where he is involved in PCI design implementation, strategic partners liaison, and programmable logic application consultations. He holds BS degree in electrical engineering and computer science from University of California at Berkeley.

#### ***REFERENCE***

PCI Specification Rev. 2.1  
Altera Application Brief 143: Understanding FLEX 8000 Timing  
Altera Application Brief 100: Understanding Classic, MAX 7000, & MAX 9000 Timing  
Altera 1995 Databook, FLEX8000 Programmable Logic Device Family  
Altera 1995 Databook, MAX 7000E Programmable Logic Device Family

## A NEW FPGA FAMILY FOR PCI INTERFACE DESIGNS

Brian Small, QuickLogic Customer Engineering Manager  
2933 Bunker Hill Lane  
Santa Clara, CA 95054  
(408) 987-2003  
email: small@quick.mhs.compuserve.com

### **Abstract**

The flexibility of FPGAs has made them popular for implementing interfaces for buses such as the Peripheral Component Interconnect (PCI) bus. However, the PCI timing and electrical specifications are difficult for most FPGAs to meet, even for many of the FPGAs advertised as "PCI-compliant". This paper identifies and explains the most critical PCI specifications and how they relate to FPGA-based designs. It also explains how the features of QuickLogic's new pASIC 2 FPGAs allow them to meet these critical specifications, and why these devices are often the best choice for FPGA-based PCI designs.

### **PCI/FPGA Design Requirements**

To design a PCI interface into an FPGA, the design/FPGA combination must provide a set of minimal criteria. This set includes, but is not necessarily limited to:

- PCI Electrical Compliance on I/O pins attached directly to the bus
- PCI Timing Requirements
  - 7 ns setup time
  - 11 ns clock-to-output time
  - 33 MHz clock frequency
- Density and pinout to support master/target designs and back-end logic
- JTAG support (optional in PCI Specification)

These criteria will be addressed in the following sections. Also, it will be demonstrated how the QuickLogic pASIC 2 family FPGAs meet these criteria.

### **PCI Electrical Compliance for I/O Pins**

Pins which attach directly to the PCI bus must have electrical characteristics which are somewhat unique to PCI. Because the PCI bus is unterminated, and relies on reflected signal propagation to provide a strong signal to a load on the bus, all pins on the bus must meet AC switching requirements as well as DC drive requirements. These requirements are best illustrated with a Voltage vs Current diagram, showing the compliance region falling within an area of the graph. Below are shown the VI diagrams for low to high (IOH) and high to low (IOL) transitions.

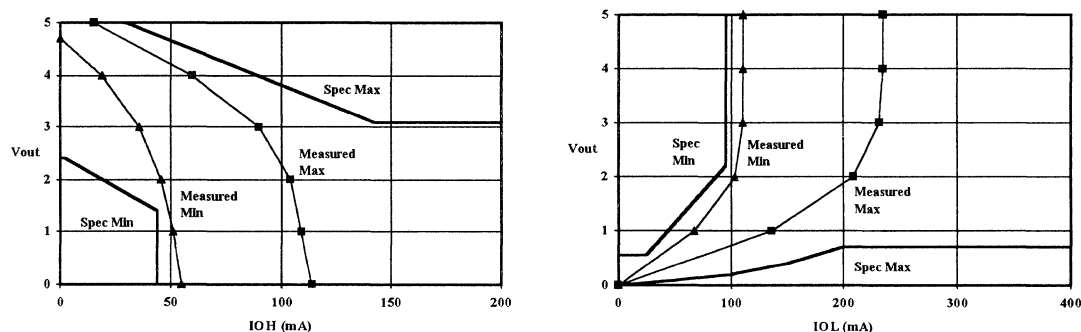


Figure 1: PCI I/O Buffer Electrical Requirements

The QuickLogic pASIC 2 I/O buffer performance is shown as dotted lines (Measured Min/Max) in the above diagrams. This data demonstrates the results of careful design of the I/O buffers for this FPGA family to meet PCI electrical requirements.

### Timing Requirements (Setup, Clock-to-Output, and Frequency)

A FPGA device should not claim to be PCI compliant without proof of timing compliance. This requires a reference design that illustrates that the device is capable of meeting the 7 ns setup times, 11 ns clock to output times, 33 MHz frequency, etc... This section will discuss the more critical problems with meeting these PCI timing requirements.

#### Setup Time

The PCI bus specification requires that all signals must be registered by the destination device within 7 ns of the rising edge of the clock. This equates to a 7 ns setup time on the FPGA. For a PCI *Target Interface* design, this is not a difficult requirement, because the PCI signals can be registered immediately upon entering the device - so there is little or no logic between the input pin and the flip-flop. The Quicklogic pASIC 2 FPGA family offers input registers on all I/O pins for this purpose. These input registers are useful for the PCI Address and Byte enable signals. By using the input flip-flops in the PAD cells, more flip-flops and logic cells are free in the internal array.

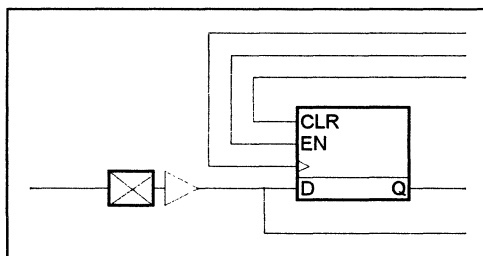


Figure 2: pASIC 2 Family Input Register

In the case of a PCI *Master/Target Interface* design, the 7 ns setup time requirement is a little more difficult to meet, because some of the PCI control signals must be interpreted and an output control signal needs to change on the same clock edge. This means that there must be logic between the input pin and the flip-flop, so input registers as in Figure 2 will not be adequate for these control pins. An example of one of the critical PCI output control signals that needs to respond to a change in the input control signals in the next clock cycle is IRDY (the signal that indicates that the master device is ready to receive/transmit data).

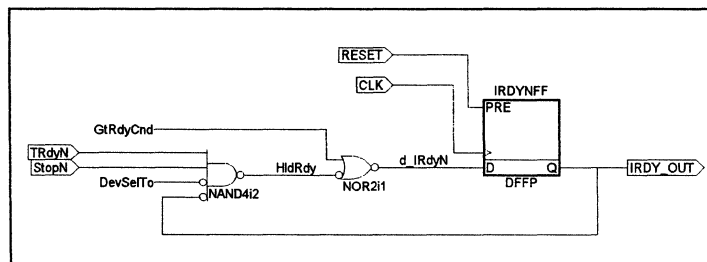
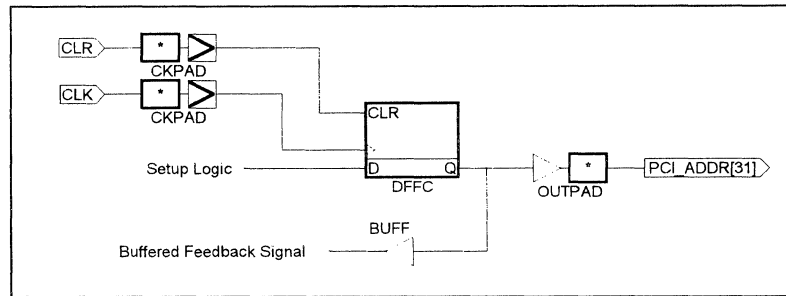


Figure 3: Example of SETUP logic for IRDY signal in PCI Master

The example in Figure 3 of the IRDY setup logic comes from the PCI Master/Target Interface Applications design that QuickLogic includes in their PCI Design Kit. Other designs may implement the logic with slight differences, but the point to be made is the same. IRDY must change based on the previous clock's STOP and TRDY signals for a proper PCI master implementation. This means that the setup time includes the logic necessary to interpret the TRDY and STOP signals on the PCI bus. The pASIC 2 architecture can easily implement this logic such that the path from TRDY and STOP to the IRDY flip-flop takes only one logic cell delay. When this total delay from TRDY or STOP to the IRDY flip flop is calculated, the minimum CLK pin to IRDY flip-flop delay is subtracted to determine the SETUP time. This falls well under the 7ns setup time requirement of the PCI specification, because of the versatility of the pASIC 2 Logic Cell, combined with fast routing.

## Clock-to-Output

The PCI specification requires that all signals driving the PCI bus must be available to the output pins of the device within 11 ns of the rising edge of the clock. This timing is based on a 50 pF load on the output pin.



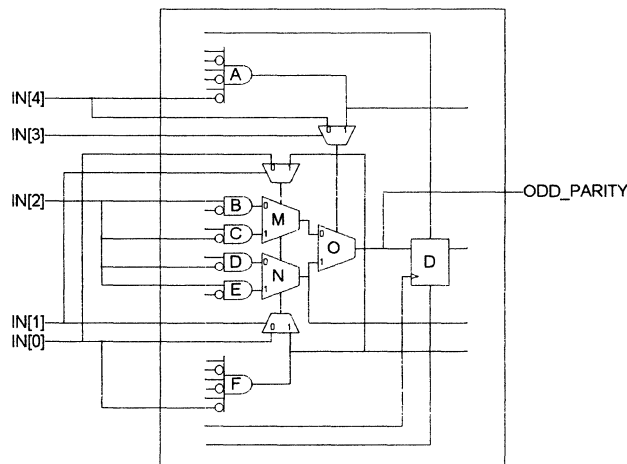
**Figure 4: Achieving 9 ns Clock-Output Delay in the pASIC 2 Architecture**

Figure 4 uses a conceptual schematic drawing to demonstrate how a 9 ns clock to output delay can be achieved in QuickLogic's pASIC 2 programmable device architecture. Notice that the fanout on the output of the flip-flop that drives the output pin is minimized by buffering the feedback path. This approach limits the fanout on the output of the flip-flop to 2 loads (the pin and the feedback buffer). In the pASIC 2 architecture, the Clock to flip-flop delay is fixed through the clock network at about 4 ns. The delay from the flip-flop output through the wire with a fanout of 2, and to the output pin with a 50 pF load, is about 5 ns. These numbers add up to a 9 ns clk-to-out time, well within the PCI requirement of 11 ns.

In some FPGA architectures, it is necessary to include a flip-flop in the output pad in order to reduce the delay from the flip-flop to the output pin. However, in these cases the same flip-flop must be duplicated in the internal array in order to provide a registered feedback signal. QuickLogic's pASIC 2 architecture permits the use of the internal array flip-flops for fast clock-to-output delays, making flip-flops in the output cell unnecessary.

## Frequency

The PCI specification requires a PCI interface device to run at up to 33 MHz. Since most PCI board designs run at the maximum 33 MHz, then an FPGA PCI interface should be able to function at this maximum speed to be considered PCI compliant. One example of a critical path in a PCI interface is the 37-bit parity circuit, which must produce even parity across the 32-bit PCI data bus, the 4 PCI byte enables, and the PARITY input signal in one 33 MHz cycle (30 ns). Figure 5 illustrates how a pASIC 2 device can implement a 5-input parity function (equivalent to a five input XOR gate) in a single logic cell. A 37 bit parity function requires just three levels of these 5-bit parity blocks.



**Figure 5: pASIC 2 Logic Cell Showing Implementation of a 5-input Parity Function**

The Logic Optimizer Tool, built into the pASIC 2 place and route software automatically maps XOR functions efficiently from synthesis tools or schematics into the pASIC 2 logic cell. Therefore, the designer does not need to understand the complexity of the pASIC 2 logic cell to benefit from it. Using QuickLogic's pASIC 2 FPGAs, the 37-

bit parity circuit takes only 9.5 logic cells, and the critical path goes through only 3 logic cells. In a “-2” speed grade device (the fastest grade), the delay through this circuit is about 12 ns. This is only a little more than 1/3 of the 30 ns clock period allowed in a PCI interface design.

### ***Density and Pinout***

PCI designs vary widely in their requirement for pins and gate density, but this paper will include some general guidelines based on experience with real-world PCI designs.

PCI Target-only designs require only 47 pins for the PCI interface, plus 5 pins for JTAG (optional), plus whatever pins are necessary for the back-end datapath and control. A typical back-end interface for PCI Target designs includes a 32-bit address bus, 32-bit data bus, 4 byte enables, and no more than 10 control signals. This adds up to a need for about 125 pins on the target device, not including VCC, GND, and JTAG interface pins. As far as logic density is concerned, many PCI Target designs are implemented in 4000-gate FPGAs, with more complex Target interfaces seen sometimes in 6000- to 8000-gate devices. Therefore, the appropriate pASIC 2 devices include the 5000-gate QL2005 and the 7000-gate QL2007.

PCI Master/Target designs are more complex than PCI Target-only designs. Typically, some sort of DMA controller resides on the device as well as the Master/Target Interface and the back-end interface logic. Also, the minimum pinout on the PCI end increases to 49. For these applications, the 7000-gate QL2007 and 9000-gate QL2009 are most appropriate. The density and pinout information is shown in Table 1.

**Table 1: QuickLogic pASIC 2 PCI Devices**

<b>pASIC 2 Device</b>	<b>Density</b>	<b>User Pins</b>	<b>JTAG?</b>	<b>Appropriate PCI Interface</b>
QL2005	5000 usable gates	156	Yes	Target-Only
QL2007	7000 usable gates	192	Yes	Target-Only or Master/Target
QL2009	9000 usable gates	228	Yes	Master/Target

### ***JTAG Support***

The PCI 2.1 specification indicates that JTAG (IEEE standard 1149.1) is an optional component of a PCI interface design. The JTAG port on an FPGA consists of either 4 or 5 pins which are used on the board only (not across the PCI bus) for the purpose of verifying the pin connections of each device to the board and the interconnection between devices. All QuickLogic pASIC 2 devices have a 5-port JTAG interface. All I/O pins can be loaded, read, and enabled with the JTAG interface.

### ***Conclusion***

PCI Interface designs offer many new challenges for FPGA designers. A designer needs to look carefully before making a decision about which FPGA offers the best solution. QuickLogic’s pASIC 2 FPGA devices are designed to meet the needs of many new high-speed, high-density, and high-pinout designs - including PCI. Aspects of the device architecture such as input registers, flexible logic cells, high speed routing, PCI I/O buffer compliance, and a JTAG interface makes it possible to design PCI interfaces in the shortest possible development time.

## **PCI Implementation Kits for ORCA FPGAs: Features and Design Considerations**

James F. Hoff  
Lucent Technologies  
555 Union Boulevard  
Allentown, PA 18103-1229  
800-372-2447/610-712-4666 (fax)  
e-mail: hoff@lucent.com

*This paper discusses the three components which have provided designers with a way of generating high-speed bus designs in very short times. A particular product, Lucent Technologies' ORCA PCI Kit, is presented which provides some distinct advantages for the designer, and some issues are discussed regarding its implementation.*

A quantum leap in computer high-speed bus architecture functionality is now occurring, enabled by advancements in several related areas. The first advancement is the emergence of a bus specification that has enough proponents to become a de facto standard — the PCI Bus. This bus architecture has achieved this status by being fast, well defined, and by having an inexpensive direct-to-silicon interface. The second advancement is the migration to Hardware Description Languages (HDLs). HDLs have enabled logic designers to generate huge quantities of complex logic in a short time by breaking a design down into a text-based hierarchical tree of reusable modules. The third advancement is the FPGA, which has revolutionized logic design by providing a design medium somewhere between discrete logic and the traditional ASIC and which provides the advantages of both in many applications, plus some uniquely its own.

These three catalysts have ignited an explosion of new designs. The HDL-based PCI Bus on an FPGA puts complex, flexible, quick-turn bus designs within the reach of a wide range of potential applications. What is more, the designer only needs to deal with issues specific to a particular design, with little duplicated effort.

This paper discusses some of the characteristics of this new bus design implementation medium, and then goes on to explain in more detail an implementation based on Lucent Technologies' (formerly AT&T Microelectronics') ORCA FPGA. The rationale for choosing this FPGA is presented. Finally, some specific design issues are considered.

### **Advancement #1: The PCI Bus**

It is assumed that readers of this paper are familiar with the features of the PCI Bus; however, several characteristics of the PCI Bus have been crucial to its success as a standard in this area:

- The bus interface is specified as a single-chip interface with no "glue logic" (pull-up resistors, buffers and drivers) necessary, which is especially important in an integrated FPGA solution.
- It is fast — so fast, in fact, that it is often not the system's bottleneck to throughput, and can support bandwidth-intensive peripherals such as SCSI, LAN and video.
- The PCI Bus is flexible, allowing individual implementations to select from an extensive suite of optional features without encumbering all implementations with those features' overhead.
- Resistance to obsolescence has been designed into the standard.
- It was developed just as the need was emerging for a bus with the above features (i.e., it was timely).
- Now that it is the standard in its field, it enjoys all the benefits attendant to that position, such as familiarity and interoperability.

On the other hand, certain PCI Bus characteristics cause problems for the designer, particularly in the connector interface:

- To meet the PCI Bus' stringent timing requirements, demanding propagation delay specifications were imposed which thin out the field of prospective FPGAs considerably.
- The requirement that the PCI Bus interface directly to a single chip with no "glue logic" also means that there is no way to "fix" an FPGA with unsuitable DC, AC or parasitic characteristics.

### **Advancement #2: The Hardware Description Language (HDL)**

The format of the design medium has had a significant impact on the rate of success of PCI Bus designs, and especially FPGA-based designs, which tend to have short design cycles. Also, as designs have become very large (15,000 or more gates), it has become difficult to design with traditional design-entry methods, such as schematics and programmable-logic equations. These methods also tend to be technology-dependent, and often require the designer to make trivial low-level design decisions (e.g., muxes versus NANDs). Finally, design media such as schematics do not easily lend themselves to hierarchical techniques and modular reusability. For these and other reasons, HDLs are overtaking older methods as the standard for design entry. Two HDLs have developed more or less simultaneously, but neither seems to be emerging as the winner in the battle for acceptance. VHDL, which is of military origin, is very rich in capability and, like its sister programming language, ADA, tends to be wordy and structured. Verilog, on the other hand, is less capable and, like its sister language, C, is concise and permissive. The military/commercial distinction has since blurred, and now the difference is more a matter of region and company culture.

Regardless of which HDL is chosen, the advantages of high-level design are many. Vendors can (and do) now offer "design kits", pre-designed modules that perform complex but generic functionality. All that is necessary is to tailor the design to a specific application. Within a company, there is a high likelihood that a module designed for one project can be easily adapted for use on a subsequent project, and a design defined in HDL can be quickly converted from one implementation medium, such as FPGA, to another, such as ASIC. Quick design turn is the result, a necessity in the current marketplace. And just as designers have come to develop products with components like processors and LAN controllers with inner workings that are unfamiliar, designers can now design with kits such as PCI Bus or DSP, without extensive experience with their internals.

### **Advancement #3: The FPGA**

The FPGA has been on the digital design scene for about a decade now, but its popularity seems to be only increasing. On the low-density end, it has all but replaced discrete digital logic. On the high-density end, it is even making forays into territory formerly held by ASICs. The reasons for this are clear: the low-end SRAM-based FPGA is now as inexpensive as two or three PLAs, yet it provides a host of advantages. Among them are: functional universality; reprogrammability; remote-, MPU- or MCU-based downloading; 100% testing; protection against obsolescence; low power consumption; high speed; multiplexed functionality in a single chip; a standard CMOS interface; and invisible last-minute (or even in-field) bug fixing. The features that enable an FPGA to compete against ASICs on the high-density end are no less impressive. Once thought of only as an ASIC development tool, the FPGA is refusing to relinquish its socket. Once designed in, the design never seems to settle down enough to justify the stiff NRE and accompanying risk of error or obsolescence. Meanwhile, FPGA equivalent gate sizes continue to soar and prices to plummet, as feature geometries shrink.

### **ORCA**

This paper focuses on a particular series of FPGAs that has been designed with the PCI Bus application in mind. The ORCA 2C series, from Lucent Technologies, and its derivatives provide many features that facilitate their use in a PCI application. Among them:

- **Size** : the PCI Bus module alone requires 6k-10k gates for implementation. Thus parts with 10k-40k or more gates are necessary to accommodate the PCI bus and its back-end application. The ORCA 2C provides this.

- **Speed** : as mentioned earlier, the PCI Bus standard requires very strict clock-to-out propagation delays of only 11 nanoseconds, which the ORCA 2C can meet.
- **Compliant with other PCI interface requirements.** A host of other parametrics constrains the I/O pins on the PCI interface, since the FPGA provides the sole and direct connection to a very high-speed bus. This includes V/I relationships, slew rates, pin capacitances and inductances, leakage currents, and others.
- **Routability** : ORCA's abundant nibble-based resources are necessary not only to make routing possible, but also to make it meet the 33-MHz clock requirements.
- **RAMs** : most PCI Bus peripherals require buffering, and the ORCA provides it in the form of high-density RAMs, both asynchronous and synchronous, both single-port and dual-port. For a given equivalent-gate rating, ORCA offers larger RAMs than other FPGAs.
- **Bounteous I/O** : as with RAM size, the amount of I/O that ORCA possesses is greater than for other FPGAs claiming the same equivalent-gate rating. This is important, since both the PCI Bus interface and the back-end peripheral interface can be highly I/O intensive.
- **5V and 3.3V** : In addition to standard 5V capability (type "C"), ORCA provides 3.3V parts (type "T"), allowing the PCI Bus to utilize its 3.3V optional capability as well.
- **We own the foundry** : ORCA's benefits stem not only from its efficient architectural specification; Lucent is the only major FPGA manufacturer that owns its own foundries. As such, Lucent has consistently provided denser, faster, lower-power parts to its customers by being first to migrate to ever-finer gate geometries. Presently, while others are announcing migration to 0.5 micron feature sizes, Lucent has devices available today in 0.35 micron sizes. In addition, in-house foundries allow for greater control over production scheduling, priorities, and quality.
- **Design kits** : Lucent has provided two design kits, called Configurable Solution Cores (CSCs), specifically for PCI Bus use: one for initiator applications, and one for target applications.

### ORCA's PCI Design Kits

Design kits make the job of integrating a complex logic entity into an overall design much easier. For the most part, the PCI Bus interface is well-defined, and special needs can be accommodated using PCI optional features that provide variations on the main theme. The PCI design kits offered by Lucent provide enough power and flexibility to meet most any need. Here are some of its features:

- Separate kits are available for initiator and target applications.
- The kits take advantage of the power of HDL logic definition. Both VHDL and Verilog source code are available.
- Multiple design flows for synthesis are supported: Synopsys and Exemplar Galileo.
- Workstation and PC platforms are supported.

The kits provide all necessary functionality as called for in Revision 2.1 of the PCI Local Bus Specification; in addition, many optional features are either incorporated or easily added to the design.

Some of the incorporated optional features include:

- Full 33 MHz clock speed;
- Full 32-bit I/O and memory address spaces;
- 64-bit address/data bus on master;
- Burst Mode support (zero-wait state in some cases);
- Retry, Disconnect and Target-Abort;
- 3.3V operation (using "T"-series parts; no software impact)
- Parity generation/checking (usually required);
- Single interrupt capability;
- Subsystem ID and Subsystem Vendor ID;
- Latency timer on master;
- Min\_Gnt and Max\_Lat registers on master;
- 16 X 64 SRAM buffer on master's local peripheral interface.



Features that can be added to the design as extensions, by modification of the HDL code, include:

- 64-bit address/data bus extension on target or 32-bit reduction on the master.
- Multiple interrupt capability may be added to support more complex interrupt schemes.
- Special Cycle. This is useful if sideband control signals are needed for inter-bus agent communication.
- Delayed transactions. This allows a PCI read to be handled in two parts, one to initiate the data fetch in the target, and the other to effect the data transfer over the PCI bus.
- Exclusive access (“LOCK#”). This may be useful for bridge applications.
- Address stepping can be added to reduce noise on the PCI bus.
- An asynchronous interface to the local peripheral allows the peripheral to operate from a different clock than the PCI Bus’ 33 MHz (or other) clock.
- PCI-side controlled bus master. Some local applications do not have a processor to initiate data transfers. Rather than add a processor, it may be more practical to give another PCI master that task. This can be achieved by making the local address and command registers accessible over the PCI bus by adding them to the configuration register space.
- Modification of SRAM buffer. The implemented scheme may provide more, less, or different capability than the local peripheral requires. For example, some applications may not require the SRAM buffer at all. Others may require a dual-port FIFO which supports uninterrupted data bursts of any length.

This does not imply that implementation code exists for all of the above features, but rather that the supplied HDL code provides a good starting point from which to design them in.

### **Special Design Considerations**

A major advantage of specifying a design in HDL is its technology-independence. However, this can become an impediment when it is important to extract every bit of performance from a logic family in order to meet demanding requirements. This is the case with the PCI Bus’ I/O interface. In order to meet the 11-ns clock-to-output propagation delay requirement, the paths from registers to output pads must use special direct-out routing. To accomplish this, Lucent’s PCI Design Kits for ORCA supply a “preference file” that contains the necessary predefined register placements and routing priorities.

Another especially demanding portion of the PCI design is the control logic. Here, signals such as “trdy#” must be evaluated as a function of large numbers of other signals. To achieve this within the confines of a 30-nanosecond clock period requires maximum utilization of the ORCA PFU’s capability of handling functions of up to eleven variables. For this reason, “trdy#” and several other key signal decodes are provided in the design kit as precompiled macros.

In addition to the special requirements above, control of the compilation process and analysis of the results is provided by constraints specified in the preference file. This includes the clock speed of 33 MHz and all necessary propagation delay, setup, and hold time requirements.

### **Back-End Interface Design**

The back-end interface is one place where the end-user will surely need to customize the supplied HDL code. Although a standard interface is defined, individual requirements will generally require modifications. Nevertheless, the functionality supplied and protocol used will make the fitting process as painless as possible.

Lucent’s two PCI implementation kits, master and target, have significantly different local-side interfaces, which is necessary because the origin of all control in a PCI interchange is the master’s local peripheral logic. The master’s local peripheral controls the master’s PCI logic, which controls the target’s PCI logic, which controls the target’s local peripheral. The following paragraphs describe the target and master blocks and their local interfaces.



### Figure 1: Block Diagram of PCI Bus Master

The PCI Bus master block diagram is shown in Figure 1. Its local interface is more complex for two reasons; first, it must be capable of initiating commands, and second, it includes a 16x64 buffer. The local interface signal pins are listed below:

- resetR#           OUT    LO = re-clocked PCI reset
- irq                IN     HI = interrupt request
- clkout            OUT    buffered PCI clock
- ldi[63:0]         IN     input data bus
- ldoR[63:0]        OUT    output data bus
- ramreg#           IN     HI = RAM buffer, LO = register
- adrcom#           IN     HI = address register, LO = command register
- rdwr#             IN     HI = read command, LO = write command
- strobe            IN     HI = command strobe
- masabrt           OUT    HI = PCI master abort occurred
- trgabrt           OUT    HI = PCI target abort occurred
- pcibusyr          OUT    HI = PCI master busy transferring data
- xfrdoneR          OUT    HI = PCI data transfer complete

All data transfers are initiated by the local peripheral side. Data can be transferred over the PCI bus in burst mode to/from the 16x64 buffer. To initiate a PCI bus transfer, the local peripheral supplies an address and a command word to the PCI block. The command word contains a transfer count and a read/write bit, plus other bits. When the command register is written, a read or write is initiated, and “pcibusyR” is asserted. If more than sixteen words are to be transferred, the PCI transaction is ended and “pcibusyR” is de-asserted, signaling to the local peripheral to supply the next data block. The PCI block will automatically update its address and restart the PCI bus transfer when it receives the next sixteen words.

#### Summary

A PCI design kit for FPGAs that is defined in HDL allows the designer to define, build and test a bus-based peripheral in minimum time. Lucent Technologies’ ORCA PCI Configurable Solution Core offers many unique features to enhance performance and reduce design time.

MPEG BRIDGES USING T1 LINES  
Tom Thorsteinson  
Linear Systems Ltd.  
959 Powell Ave.  
Winnipeg, MB Canada, R3H 0H4  
Ph (204) 632-4300, Fx (204) 697-2417  
e-mail: thor@magic.mb.ca

## ***ABSTRACT***

MPEG Video has become the industry standard method of digitally encoding moving pictures into a compressed bit stream. Once converted, MPEG data can be transmitted between computers using T1 lines before being stored or decompressed for display. The match is very close between the T1 rate, which is 1.544 Mbits/s and MPEG 1 video encoded to give comparable quality to a VCR. This paper will discuss the MPEG bit stream and how it may be transmitted as a series of frames as used by data communications protocols. Each stage of the process will be described, including potential problems due to errors caused by transmission noise. The serial communication card used to convert the serial MPEG stream into frames of data will be described along with the CSU/DSU used to convert the serial data stream into the time division multiplexed T1 stream.

## ***THE COMPONENTS OF A T1 BRIDGE***

Let us assume that we need to build a bridge which will connect two computers, A and B. For the sake of our discussion we will assume that they are ISA or PCI bus computers, of 486 or Pentium class, that is, the sort of computers which are commonly used for multimedia applications. We wish to link A and B with a full duplex T1 Bridge so that A can transmit to B and B can also transmit to A, simultaneously.

We also assume that we have a source of MPEG digital program material at each end. This program material can be from a live camera, or from a hard disk or CD ROM which contains stored program material. Whether live or from memory, this program material must be encoded at a rate which the T1 link can support, that is, somewhat less than 1.544 Mbits/sec. This is a limiting factor. The program material is buffered into system RAM memory and then directed to the serial communications card. The card performs a parallel to serial conversion and sends the data to the CSU/DSU. The CSU/DSU takes the serial data stream and multiplexes it into Extended Super Frame (ESF) format which is compatible with North American T1 switches. The serial data is transmitted over the T1 line to the central office where it is sent over the telephone system to its destination. When it arrives at the destination it demultiplexed and is converted to its original form as an MPEG data stream ready to be decompressed by a decoder and then displayed or stored for future use. Because the T1 service is full duplex, there can be two unrelated MPEG streams transmitted at the same time in opposite directions.

## ***BIT RATE, A CONTROLLING FACTOR***

Video by its very nature is time dependent. The fact that a film or video tape is converted to a serial bit pattern doesn't change this. If it is late, it will be observed as an error. T1 will limit the bit rate to a maximum of 1.544 Mbit/s. In practice the MPEG bit rate should be lower because the T1 actually provides a clear channel of 1.536 Mbit/s for carrying data. This is because a certain amount of the T1 bandwidth is used for synchronization and control signaling.

Another loss in bit rate can be caused by the efficiency of the transmission framing used by the communications card. The communications card sends data in a series of frames. Each frame consists of a series of fields including flag, address field, control field, information (data) field and a frame check sequence (FCS) field. This overhead will further lower the actual throughput. It should be noted that you can use longer frames, say 2K bytes or so to increase efficiency by increasing the amount of data relative to the framing information. Furthermore if an error correcting protocol is used for the link, then acknowledgments frames will be returned for each frame of data. You can therefore estimate that only about 90 to 92% of the available bandwidth is used for transmitting data. Assuming a 90% efficiency, this means that with T1, you can actually send at a rate of 1.38 M bit/s

A third cause in loss of available bandwidth can be the computer system itself. If the transmitting computer is delayed in sending a frame of data because it had to service another task, running concurrently, then the

communications card will fill the missing space with flags. Flags are used by communications protocols to delineate the frames and to maintain synchronization in the absence of data. We will assume that we will use a computer with sufficient CPU cycles to handle the communications task and any other task that may have to run concurrently such as MPEG encoding and decoding.

A fourth cause of lost bit rate can be caused by the communication protocol itself when it corrects an error in the received data. If a full error correcting protocol is used such as HDLC or PPP protocols, then acknowledgments are constantly being sent back to the transmitter by the receiver saying that each frame has been received and that there are no errors. If an error is detected, then the transmitter is instructed to resend a frame. This can be very disruptive to the smooth flow of data required by MPEG 1.

MPEG compressed video is a stream digital information. Both video and audio are combined in the bit stream. The method used by MPEG-1 to compress the video will not be described here in this paper. It is a topic on it's own. We will assume that the bit rate of this stream is determined by the encoding process. Let us assume that the encoder is set to provide a nominal MPEG 1 video (352 x 240 x 30, 1.15 Mbit/s) rate. Lets forget about the fact that MPEG talks about video frames being transmitted at a rate of 30 per second, it is still a stream of 1's and 0's and there is no need to make any sense out of it until it reaches the decoder. Lets consider it to be just a stream of data with a bit rate no greater than 1.15 Mbit/s. At this rate we can see that there should be no problems in sending it over a T1 line through our computer system, if the transmission system has a capacity of 1.38 Mbit/s.

### ***THE CSU/DSU***

The Channel Service Unit/Digital Service Unit (CSU/DSU) is a fairly complex piece of equipment. It takes the serial data stream, for example our MPEG 1 bit stream, and converts it to a format that can be accepted by the DSX-1 interface for T1 required by the telephone company. In a sense it acts like a digital modem.

We will describe the CSU/DSU as it is used to format the MPEG data stream into the Extended Super Frame (ESF) format. This is one of the frame formats required by the communications switches in the telephone company central offices. We have chosen ESF framing because it can be used to transmit a continuous stream of data at the maximum rate possible through a T1 line.

The ESF frame consists of 24 eight bit words, or time slots transmitted at a speed of 1.544 Mbit/s. The combination of 24 time slots produces a data rate of 1.536 Mbit/s. Each time slot has a framing bit associated with it. The eight bit word along with it's associated framing bit is called a D4 frame. The framing bit is used to maintain synchronization and in effect provides a timing clock for decoding the data at the receiver. The extra framing bit adds an extra 8,000 bit/s for a total bit rate of 1.544 Mbit/s.

It is an interesting aside that by using Pulse Code Modulation (PCM) to encode the voice digitally, that 8 bits of resolution gives a reasonable signal to noise ratio for telephone applications. It could have been 7 bits or 9 bits, and then we would not have the symmetry between computer bytes and the D4 frame.

The 24 framing bits which are contained in the ESF frame are not all needed for synchronization and therefore only every fourth D4 frame (4, 8, 12, 16, 20 and 24) in the superframe is used for synchronization. The result is that 6,000 bit/s can be used for other functions. This is used for performing continuous error checking on the ESF frame and for signaling back to the central office. This signaling is called the facilities data link (FDL) and is used by the phone company to test the operation of T1 circuits with diagnostics such as loop backs and statistics. The net result of this is that all CSU/DSUs which support ESF are capable of testing and storing 24 hours of statistics on T1 line quality. This can be a source of useful information about line quality if it becomes an issue in your application.

The frame synchronization which the CSU/DSU decodes from the T1 data stream is used to create a local clock. In a sense, the clock rate is determined by the central office. This is a precise frequency, accurate to +/- 50 ppm accuracy and is used by the communications card as a reference clock frequency when it sends data to and receives data from the CSU/DSU.

The CSU/DSU can appear in several forms. It may be a small box, about the size of a modem which sits on the desk top and is powered by a battery eliminator. It may also be a larger piece of rack mounted equipment which requires a 115V power supply. A third incarnation may be as a small module or daughter board which plugs into the

serial communications card in your computer, or as an integral part of the communications card. We have experience with both the small box and the daughter board form which we have developed and call the IM CSU.

### ***THE SERIAL COMMUNICATIONS CARD***

Our experience with using communications cards for compressed video applications lies with two different PC designs. The first is called the T1 Master which uses an ISA bus and the second is called the WAN Master which uses a PCI bus. Both designs can be used in 486 and Pentium class personal computers, to provide two serial interfaces at T1 rates.

#### **The T1 Master**

The T1 Master has two channels and has an ISA bus interface which is capable of an aggregate data rate of approximately 6 M bit/s. It uses the Zilog 16C32 IUSC Serial Communications Controller, which is also known as the IUSC. The performance of the card is an excellent match for T1 providing the computer is not busy with other tasks other than the main communications task. The T1 Master is a Bus Master DMA design. It does not have a large internal data buffer or an on board processor. It uses a very high performance DMA which is built into the IUSC which also cascades with the host system DMA. By selecting the size of the circular queues, it is easy to keep the data buffers on the IUSC serviced in spite of latencies which may appear in the operating system when other tasks are running, or when very high priority tasks are serviced. The size of the queues is limited only by the size of the system memory.

The FIFO memory in each of the IUSC is therefore only necessary to compensate for the very short time delays due to hardware latency. For example, when the T1 Master does a DMA request, there will be a variable short time before it is available. This is easily accommodated by the relatively small FIFO buffer in the IUSC.

The DMA can address the full 16 Mbytes of ISA system memory and can be used effectively in Linked List DMA mode. Using this mode, the software driver can utilize circular data queues of arbitrary size to store data for transmit and receive. The queues can be designed to circulate without interrupting the host processor. Data can therefore be transmitted and received without host processor intervention. The host processor must only get involved when data is moved or protocol processing must be done. The ultimate speed of the T1 Master appears to be limited by the latency and clocking rate of the host system DMA controller. The ISA bus is limited to a clock rate of 8 MHz. This can be adjusted higher in some system BIOS options, but the higher rate is non standard and not always supported by all cards in the system.

#### **The WAN Master**

The Wan Master is a further development of the T1 Master. It eliminates the bottleneck to data flow caused by the ISA bus DMA and provides a plug and play environment for interrupt and address selection. The WAN Master is also a two channel board which uses the same IUSC as the T1 Master. This makes porting software easier between the two boards. The WAN Master uses a slightly different interface philosophy in that it has 256 Kbytes of on board buffer memory. The advanced DMA modes of the IUSC can transfer large amounts of transmit and receive data between the buffer memory and the IUSCs without host intervention. The data is buffered by the same type of circular queues in this memory area except that there is no need to cascade DMA controllers. The IUSC DMA controllers can now run at maximum rates and the DMA request latency is eliminated.

The host system has direct access to the buffer memory through the PCI bridge interface. Data can be transferred in two ways. One method uses standard memory block move instructions which are under processor control. This is adequate in many applications. However, for applications where the processor may be busy, a DMA controller is available in the PCI bridge chip to move data. The result of the high performance PCI bus interface is that the IUSC can operate at much higher rates. The WAN Master supports serial data rates of up to 10 Mbit/s (full duplex) on both channels simultaneously. Using high speed interfaces such as HSSI, the WAN Master can support data rates of up to 20 Mbit/s full duplex. These data rates are high enough to drive MPEG 2 video applications to the fullest extent.

The large bus bandwidth of the PCI interface can be used to its fullest potential when there is a need for multiple WAN Master cards in one host system. Installing four WAN Masters in a system can create a throughput of 80 Mbit/sec, which is less than 7% of the 132 Mbyte/sec potential of the bus.

## **SOFTWARE DRIVERS**

There are two approaches which we use to provide a driver interface to the communications cards. One is to use a standardized structured approach such as a network driver. Options include Network Driver Interface Specification, (NDIS) driver, the Open Data Link Interface (ODI), and the Crynwr Packet Driver for MS-DOS and Windows 3.1. In our example we will concentrate on the NDIS driver and TCP/IP (PPP) or NetBEUI, with which we have the most experience in testing.

The NDIS specification was written by Microsoft and 3Com to provide a shareable device interface for LAN Manager. ODI is a Novell standard (Open Data-Link Interface) which allows other protocols including TCP/IP (PPP) to work simultaneously with NetWare. NDIS and ODI can support multiple protocol stacks. The Packet Driver is a small and relatively simple interface and has the potential for the highest efficiency.

While there is a choice of protocols which can be used for the data link layer, TCP/IP (PPP) appears to be a good solution especially in comparison to the well-proven HDLC protocol. HDLC is very good for transmitting data with a very high accuracy at the data link level. It acknowledges each frame and checks the sequence of frames received continuously. In addition, a sliding window allows only a small number of frames to be outstanding without acknowledgment. If an error is detected, it is corrected in the data link layer. For noisy analog lines where the probability of an error is high and data rates are relatively low, this is a good solution. However, when sending data over high quality links such as a satellite or fiber optic links, bit error rates as low as  $10^{-10}$  are typical. Also, given the fact that compressed video is rather forgiving to the occasional error, it then becomes feasible to use a link layer protocol which is optimized more for speed and efficiency. With this error rate, an error may crop up once every two hours or so.

PPP is therefore a good protocol to use for MPEG Video since the frames are transmitted without requiring frame by frame acknowledgments at the link layer. This means that the link layer software can be designed with large DMA ring buffers which can be filled and transmitted or received continuously. This improves the efficiency of the driver. When an error correction is required it is handled by the transport layer which is TCP.

The most simple, direct and obvious way to move the MPEG data between two computers is to use the file system which is built into the operating system. If you are using DOS or Windows 3.1, the NDIS 2 driver may be loaded and a transport protocol such as NetBEUI or TCP/IP (PPP) can be bound to it. The file system approach gives acceptable results with a fairly powerful computer such as a 486/100 or Pentium 120 operating on the client end, but may have some problems maintaining the data flow rate on the server system. The problem appears as a jerky image with gaps in the sound. If you are using the Windows NT operating system on the server the file system seems to be much more efficient and can give smooth video and sound on the Client. As an additional note, generally the Client computer has a heavier load because of the MPEG decompression process and it also benefits from having a powerful CPU.

### **Is an Error Correcting Protocol Needed**

If the only function of the T1 link is to transmit compressed video then some consideration has to be given to whether or not the extra overhead of an error correcting protocol is needed. When an error is detected, the upper layers of the protocol will request that the data be resent. This can cause a noticeable glitch in a video stream. If there are no critical files, programs, etc. to be transmitted and the bit error rate of the communications link is of the order of  $10^{-10}$  then it is possible to avoid using any protocol and to simply "stream" the data from server to client. It all depends on the particular application. This bit error rate may produce a small glitch which may or may not be noticed over a period of several hours running at T1 speeds. If the error occurs in a data field of the MPEG frames then it may not even be noticed. A much more noticeable error occurs if it's in a motion prediction area or the discrete cosine transform area.

For this purpose, a character driver may be provided for the MS DOS operating system. In Windows NT, a kernel mode character driver can be used. These drivers provide the necessary buffering and interrupt handling interface to the communications card. They permit a custom application to be written which may bypass the file system and network drivers to give better real time control and better timing performance. When using this type of driver, there is no error correcting protocol. Errors may however be detected and recorded for later reference. HDLC type framing will be used with the frame size determined by the buffer size. All of the HDLC framing will be transparent to the data being transmitted. The character driver allows direct access to the hardware through Windows NT HAL (Hardware Abstraction Layer) calls whereas the NDIS driver can only talk to the HAL through the NDIS Wrapper functions.

## **ADSL Technology**

If data is transmitted in this way with no error correction then there is no need for acknowledgment in the reverse direction. The MPEG data link can then be half-duplex. This opens the possibility of using Asymmetrical Digital Subscriber Loop (ADSL) technology. With ADSL, only a single pair of the two twisted pairs of T1 cable are used for point to point connections. The data is further encoded using proprietary techniques to provide a 1.544 Mbit/sec forward channel and a 16 Kbit/s full duplex control channel. In addition, a plain old telephone (POTS) connection is provided for an analog phone. Variations on ADSL are available which will give much higher rates than a standard T1.

## ***SUMMARY***

The MPEG-1 video format is a good match with T1 lines and is a practical way to transport VCR quality video images with CD quality stereo sound. A system can be assembled using PC class computers and off the shelf board level products to transmit real time compressed video. The system can be either half or full duplex depending on the application.



# HIGH PERFORMANCE VISION PROCESSING FOR THE PCI BUS

**Fernando Serra**  
**Imaging Technology, Inc.**  
**55 Middlesex Turnpike**  
**Bedford, MA. 01730**  
**(617)275-2700 FAX: (617)275-9590**  
**e-mail: fernando@imaging.com**

## *Abstract*

This paper presents i) A design for high performance vision processing hardware for the PCI Bus ii) The use of the PCI Bus as an optimized data transfer pipeline between imaging capture/processing hardware and upcoming, high speed, general purpose CPU's. iii) What we have learned from designs on previous buses and different hardware architectures and how we have implemented some of these techniques on the PCI Bus architecture. iv) The principles behind the design of a frame grabber on the new PCI Bus and how this technology was to be applied to the follow-on designs of new vision hardware.

## *How the PCI Frame Grabber Came About*

Image capture technology within the PC market was limited by the speed at which data could be transferred between the data capture hardware and the host memory. Another limitation was the lack of unattended bus master DMA data transfers. During the period when the 386/486 and ISA/Vesa local Bus architecture dominated the market, image capture hardware for the PC was limited by the speed of the CPU and the bus. As the speed of CPUs improved the data transfer rate also improved, but the transfer speed improvement realized was not enough. The maximum data transfer speed that could be achieved was about 10 Mbytes/sec, using the Vesa Local Bus and a 486 CPU running at 66 MHz. These speeds can only support the data transfer speeds required while capturing an RS-170/CCIR 8-bit video signal. ITI provided products for the ISA, and later, VL buses and became knowledgeable about framegrabber design for the PC environment. Still, since the main processor was tasked with managing the data transfer from the framegrabber, there was not enough CPU bandwidth left for it to actually process the data. This made it almost impossible to keep up with real time image processing, which is required by a variety of vision applications. As a result, many applications had to be developed for different architectures, such as the VME bus. ITI gained considerable experience in high speed vision and image processing designs via our Modular Vision Computer (MVC) 150/40, a high performance pipeline processing architecture VME bus product which was modular in nature.

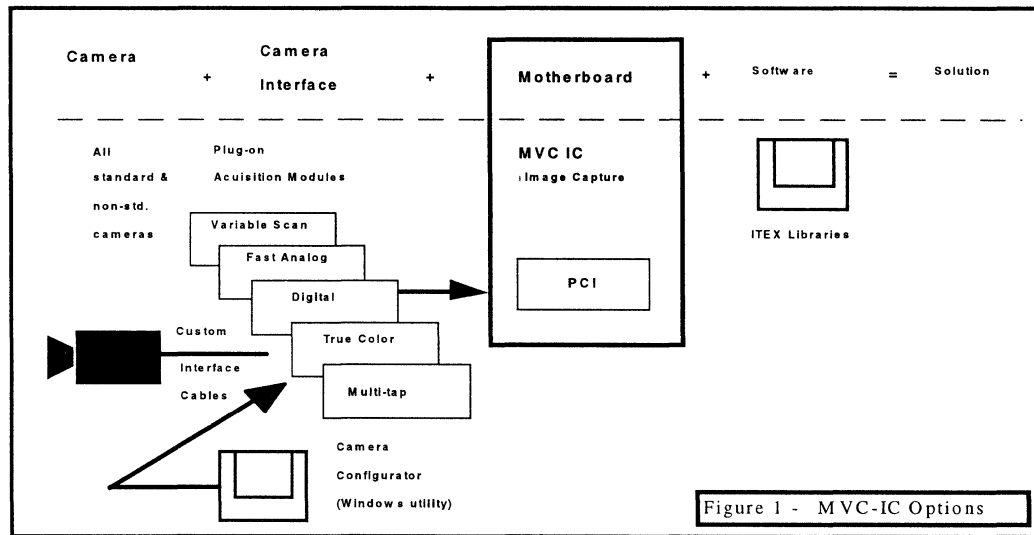
With the introduction of the PCI bus, ITI as a company looked at what had been learned from our previous products and how to apply some of this experience to the PCI Bus. We took the best features of our previous designs, like flexibility and modularity and combined these with the data transfer speed features of the PCI Bus. We had already developed an extremely flexible data input capability in our MVC 150/40 products for the VME bus, which is provided by a variety of acquisition modules. These acquisition modules give the MVC 150/40 the ability to acquire true color, variable-scan to 50 MHz, 8 to 24-bit digital and multi-tap camera formats. Figure 1 illustrates this modularity as provided by the MVC IC-PCI, one of ITI's recent PCI bus based products. The acquisition modules have been used by customers in applications that range from low end image processing systems to high speed vision systems.

ITI had also developed expertise in framegrabbers for less demanding applications and saw needs for two different PCI bus based products. One product would use the PCI bus to overcome the data transfer limitations of previous PC buses and dramatically advance the capabilities of framegrabbers within the PC architecture. The other product would bring the full performance capabilities of pipeline image and vision processing to the PCI bus

By using the input flexibility already available on the MVC 150/40, the next step was to determine the what features to implement on the framegrabber (image capture) motherboard called the MVC IC-PCI. The main goal for the design of

the IC-PCI was to transfer the data that was acquired through the acquisition module onto the PCI Bus as fast as possible. With this in mind, achieving the highest sustained speed data transfer rates possible required the framegrabber to be a PCI Bus master DMA device. Allowing the framegrabber to fully control the data transfer onto the PCI Bus or onto a slave device on the PCI Bus gave the IC-PCI a significant advantage over designs for earlier PC buses. Now the CPU was able to be available to process the data resident in memory instead of managing the data transfer from the framegrabber.

As the IC-PCI was to be DMA Bus Master device, the next step of the design was to determine what type of memory scheme to implement on the motherboard. Even though the IC-PCI was to be a DMA Bus Master device, it could be problematic for a given application to guarantee that the PCI Bus and the CPU bandwidth would always be available on demand. The need for local memory on the framegrabber became a clear requirement; this addressed the case of "what if" the acquisition input rate was faster than the processing rate. A linear memory buffer was implemented as the IC-PCI memory format, allowing the buffer to be independent of the size of the input device's resolution. The memory buffer was partitioned as a 4 Mbytes buffer (a 2 Mbytes version is also offered), making it possible to acquire data up to 4 million 8-bit pixels to accommodate newer digital cameras with resolutions of up to 2k x 2k pixels, which are becoming increasingly common.



This linear buffer memory also offers the capability to sub-divide the image buffer into smaller buffers, which makes it possible to use these smaller buffers as Ping-Pong buffers. By using Ping-Pong buffers, data can be FIFO'd inside the IC-PCI to prevent the CPU from losing frames, while processing a frame at a time. Linear memory also offers the capability of packing pixels to be transferred to a VGA (display) device; this allows the IC-PCI to transfer data to the display without requiring any pre-processing (packing the pixels). The memory was designed as dual ported memory to provide simultaneous read and write capability.

Figure 2 shows a block diagram of the IC-PCI. There are other features that were implemented inside the address generator and the PCI Bus Master controller circuitry. These features included the capability of de-interlacing data as it gets transferred to the PCI Bus from the linear buffer. A Region Of Interest ROI capability was also implemented in the design, allowing an independent rectangular region in the buffer to be transferred to the PCI the Bus.

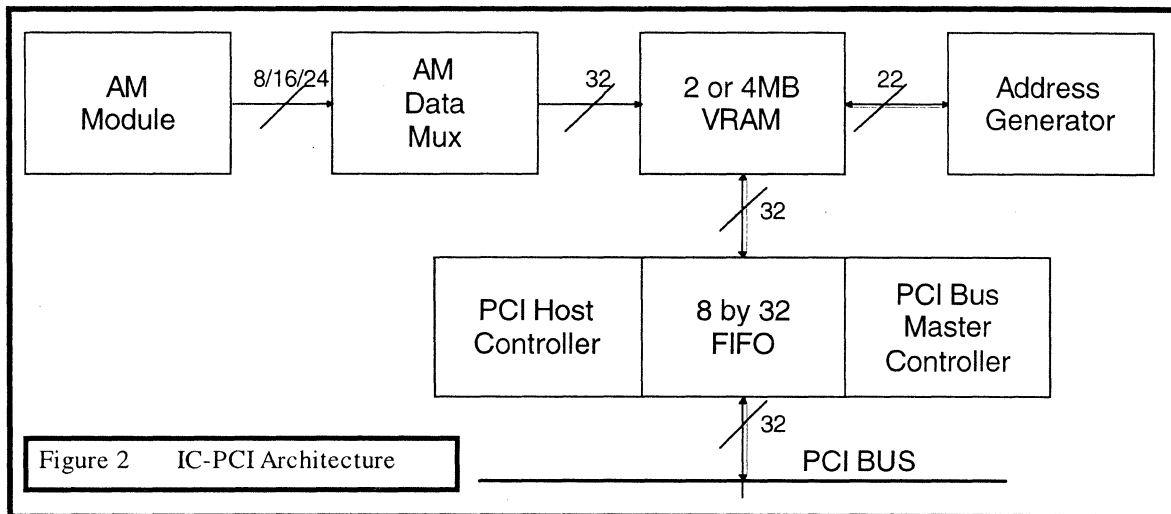


Figure 2 IC-PCI Architecture

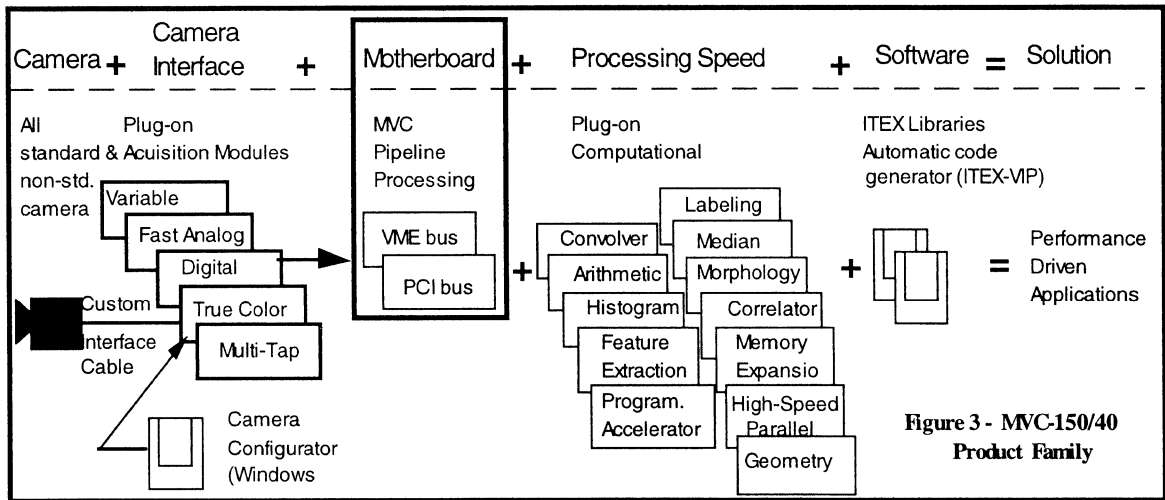
Some of the data transfer speeds achieved by using the IC-PCI in conjunction with an Intel Endeavor PCI motherboard with a 133Mhz pentium, 16 Mbytes of EDO RAM and a Number 9 Motion 771 PCI VGA card running Windows 3.11 are:

- i) A sustained data transfer of 105 Mbytes/sec between the IC-PCI and host memory.
- ii) A sustained data transfer of 76 Mbytes/sec between the IC-PCI and the VGA controller, while using DCI (Display Control Interface) protocol.

**Pipeline Vision Architecture on the PCI Bus**

While the data transfer speeds of the PCI Bus have made a substantial contribution to host based vision processing requirements, there are still cases where there is too much data for even the fastest Pentium or PentiumPro to process within the time constraints imposed by the application.. As noted, ITI has provided a pipeline architecture on the VME Bus that provides independent processing with minimal CPU involvement for these demanding applications. The PCI bus however has now made it possible for ITI to provide a combined pipeline and host based solution which can work as two independent pipelines to streamline vision applications.

Figure 3 shows the modular architecture of the MVC 150/40; the input or camera interface side of the hardware is fully compatible with the IC side of the family. On the processing side the MVC 150/40 features nine separate processing modules, making the pipeline very flexible to accommodate virtually all high speed vision algorithms.



**Figure 3 - MVC-150/40 Product Family**

The MVC-150/40 has a built-in 40 MHz 24-bit wide bus; this bus is divided into the Global Bus and the Pipeline Bus. The Global Bus is a general broadcast bus used by the Image Manager motherboard to send digital image data to any other pipeline processing boards of the MVC-150/40 family. This bus is always driven by a master device and data can be independently transferred to one or more modules at the same time. The pipeline bus is used to transfer data from one module to another. The two main buses are totally independent and they can be used either asynchronously or synchronously.

There are two kinds of PCI motherboards within the MVC 150/40 architecture. They are the Image Manager (also referred to as the IM-PCI) and the Computation Module Controller (also referred to as the CMC-PCI). The IM-PCI is the main vision device on the PCI Bus; this motherboard has the frame capture, cross-port switch, memory, timing controls, PCI DMA Bus master controller and a built-in secondary display. The CMC-PCI is a PCI Slave device; it serves as a carrier device for computational modules and also provides a cross-port switch. The main function of the cross-port switch on the CMC-PCI is to provide flexibility in the data paths into and out of the modules and the two video buses.

Located on the image managers are a number of frame buffers; these frame buffer memories are also independent in size and accessibility. The IM-PCI is offered in a variety frame buffer configurations, starting with a single frame buffer with 1 Mbyte of image buffer memory. The architectural maximum of the IM-PCI can accommodate up to 6 Mbytes of total memory where all three buffers are configured with 2 Mbytes each (although the market currently requires only 3 buffers of 1 Mbyte each, which is the current 3 buffer product offering). There is also an independent display generator which can be used to drive a secondary color or monochrome monitor in the range of RS-170/CCIR up to 1280x1024 non-interlaced VGA. The secondary monitor becomes a crucial part of many machine vision applications where the image needs to be displayed independently of the windows/menu based application residing on the system VGA. It also saves the PCI bus from being used to transfer the images for display instead of data to be processed.

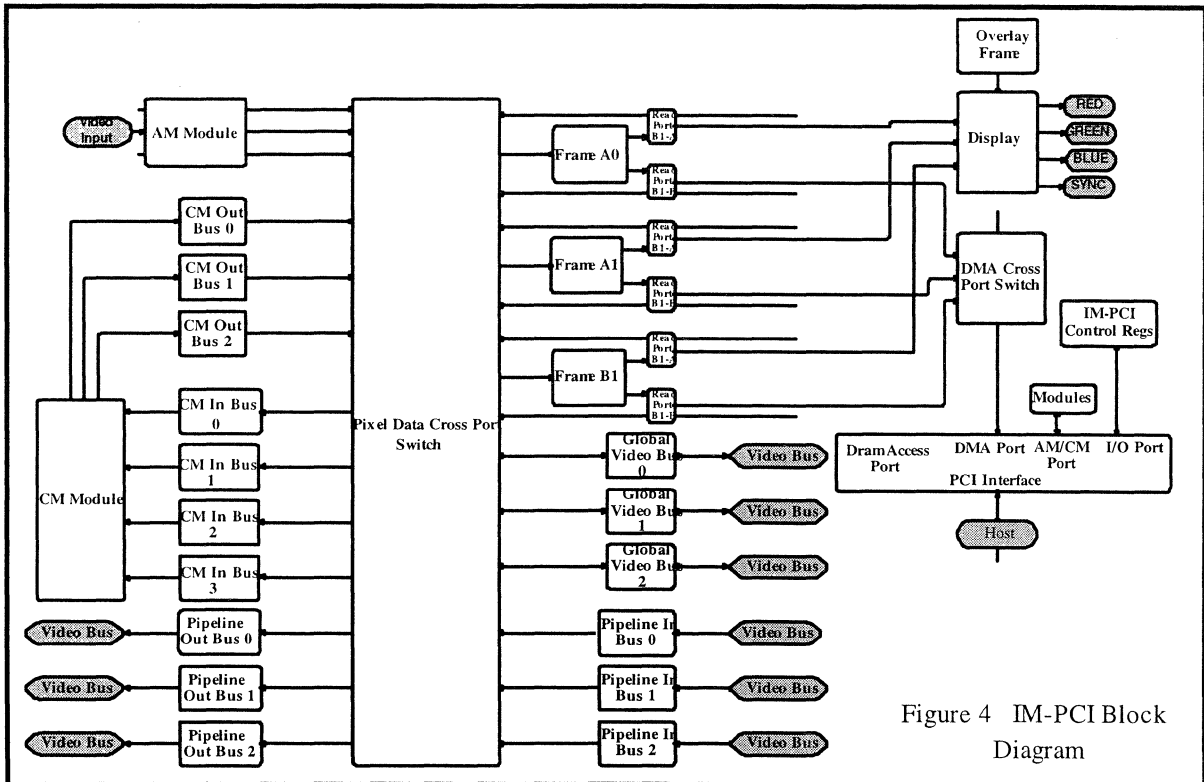


Figure 4 IM-PCI Block Diagram

Figure 4 illustrates the architecture of the Image Manager for the PCI Bus. As shown on the block diagram each memory has three ports that can be used with the internal pipeline. This feature allows the image memories to be used as two independent frame buffers, making the architecture highly flexible. As an example vision application, the same frame buffer can be used as two of the image sources required to perform a two image subtraction, placing the resulting image back into the same buffer. The secondary port is used by the DMA controller to transfer data from the frame buffer to the host via the PCI bus while using Bus Mastering. There is also a fourth port to each memory. This port is mainly used by the host CPU for host read/write functions.

The power of the MVC-150/40 pipeline architecture can be measured by the following benchmark - a 512 x 480 8-bit pixel image can be piped through any number of processing modules with the result available only 7.5 msec later. The regular frame rate for RS-170 video is about 33 msec, making it possible for the 40 MHz pipeline architecture to process four complete frames before a new capture frame comes onto the pipeline. The built-in 40 MHz AOI ( Area Of Interest ) generator enables the MVC 150/40 to have total processing independence from the incoming video timing. The MVC 150/40 can also be synchronized with the input timing, or the AOI generator can be synchronized to events generated from the input device such as vertical sync, frame reset signals, external sync, etc.

For the IM-PCI DMA controller, some features have been implemented in addition to the features carried over from the IC-PCI product design. The DMA PCI controller section of the Image Manager can also pack the pixels from the frame buffers, making it possible to directly transfer packed pixels to those VGA devices which require such data formats. The same section of the device can clip the pixel data at both ends of the 8-bit range, making the elimination of Windows 3.11 related color dots an "on-the-fly" function of the hardware. The DMA controller can sustain data transfer from the IM-PCI image memories to host memory at a rate of 66 Mbytes/sec.

## Ease-Of-Use Software For The MVC 150/40 PCI

Beyond the challenge of developing state-of-the-art high performance hardware for the PCI bus is the requirement to provide software which allows the user to exploit the full capabilities of the hardware. Further, users want the software to be easy to use as well as powerful. The task to delivering "easy to use" is quite complicated, since in the vision markets no two applications are the same. By providing OEMs and system integrators the tools required to make their product development easy and system deployment cycles shorter, ITI helps users realize the full benefit of the advanced hardware

The MVC product family is supported by a software development environment called ITEX. This is a "C" language based tool set which provides all the programmability for the Image Capture product as well as the pipeline product line. ITEX supports development under DOS, Windows 3.1.1, Windows NT, and OS2, with Windows 95 support scheduled shortly.

As part of the ITEX development environment, other tools are provided with the product. A camera configuration utility is supplied with all MVC offerings. This utility makes interfacing to complex cameras extremely simple, both for current off-the-shelf cameras and new cameras as they are introduced into the market. The camera configuration utility is a MS-Windows based tool which provides an interface to a continuously growing camera database, as well as complete access to all the MVC family acquisition modules. With the combination of ready made interface cables to most cameras in the market, the configurator makes camera interface a plug-and-play function.

Figure 5 shows an example of the ITEX camera configurator graphical interface. As the figure shows, the developer can customize with a mouse click any camera interface to meet the specific application requirements.

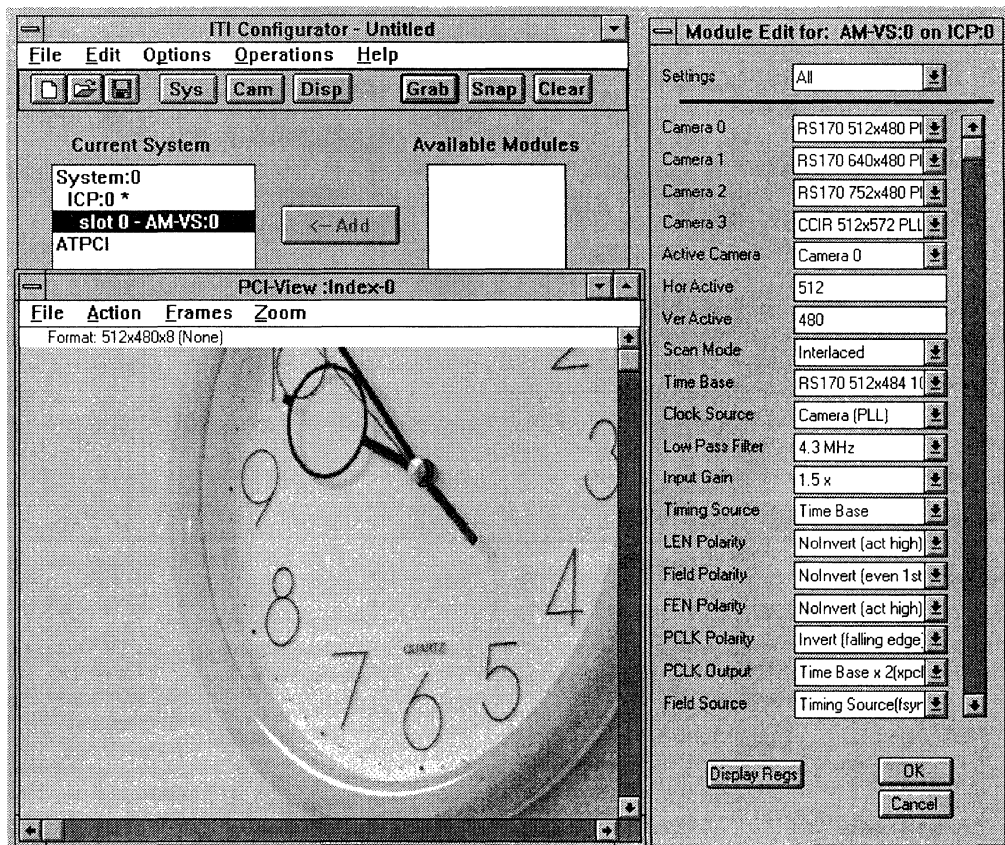


Figure 5 ITEX Camera Configuration Utility

There are a variety of third party applications software packages that range from generic image processing to bio-medical cell analysis. These packages have been developed with the MVC IC-PCI used as the data capture front-end. Now with the data transfer speeds provided by the IC-PCI, host based applications can run very close to real-time.

As part of the MVC 150/40 pipeline product, ITI has developed a graphical programming tool for the pipeline architecture called ITEX-VIP (Visual ITEX Programmer). ITEX-VIP is a Windows based “point and click” development environment, which generates well structured and optimized ITEX code. ITEX-VIP allows developers to create pipeline connections, frame buffer actions, and set-up all the computational characteristics of an algorithm to be executed on the MVC 150/40 pipeline. Figure 6 illustrates one of ITEX-VIP’s graphical representations of an MVC 150/40 motherboard. This tool generates ANSI “C” code that can be compiled under any of the platforms supported by ITEX for the MVC 150/40. ITEX-VIP is a standalone software development environment. The presence of the pipeline hardware is not required in order to develop application code. Based on a hardware configuration file, ITEX-VIP provides the developer all the flexibility to generate applications with just the use of a laptop computer running MS-Windows. This tool also has a built-in benchmark facility, which generates processing times based on the size of the image to be processed through a specified pipeline.

While ITEX-VIP is an MS-Windows application, it was designed with portability in mind. Applications developed under the MVC 150/40 PCI can be easily ported to other platforms by regenerating the data flow. ITEX code can then be generated for the MVC 150/40 VME. ITEX-VIP also provides extensive on-line context sensitive help, which provides complete information to the user about all the functionality of the MVC 150/40. VIP also allows developers to insert notes as comments as well as previously developed “C” code fragments.

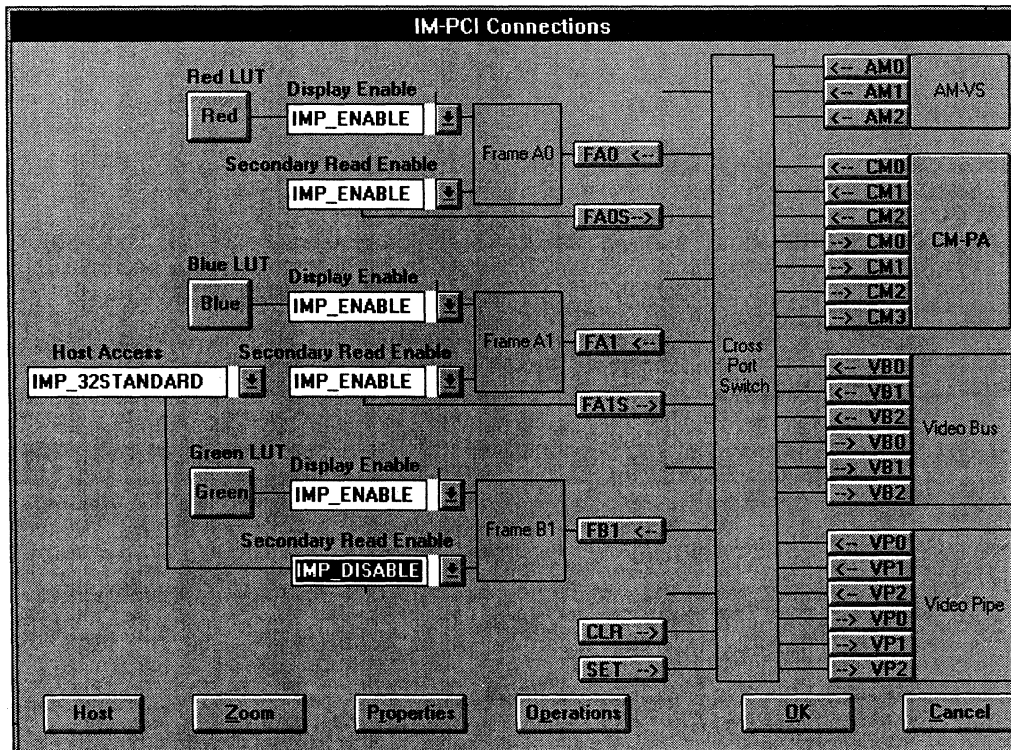


Figure 6 ITEX-VIP Graphical View of the IM-PCI motherboard

### *Summary*

ITI believes that the MVC IC-PCI and MVC 150/40 for PCI have realized the goals of bringing our high speed vision technology to the PCI Bus. The future holds developments in interfaces for new cameras, new computational modules for more flexibility and new software tools to continue improving the product's ease-of-use. We will continue the enhancement of our PCI products as the technology advances and will keep performance and flexibility as key design goals.



**THE PCI BUS  
AND  
BROADCAST QUALITY VIDEO AND AUDIO**

**presented by**

**Richard A. Kupnicki**

**LEITCH®**

**PCI**  
*LOCAL BUS*

# The PCI Bus and Broadcast Quality Video and Audio

## Abstract

Very recently, we have seen the increasing convergence of the television broadcast, telephony and computer markets. Near video on demand (NVOD), video on demand (VOD) and the associated video file servers have spurred the imagination of both manufacturers and service providers. The PCI Bus provides a mechanism which allows the processing of real-time video in uncompressed format as well as compressed. The key element in achieving this goal is the interface between the broadcast world and the PCI Bus itself.

## Introduction

This paper will provide an overview of one approach to the issues surrounding a successful interface of the technologies in question. In order to fully appreciate all of the factors involved, we will present a brief history of video compression formats with a view towards the bandwidth requirements of each and their practical uses. From here, we will move on to Leitch's approach to the PCI interface module describing the various blocks involved for each stage.

## Video Compression

It is worth noting, at the outset of this discussion, that both film and video are simply a sequence of still pictures shown at a rate which is sufficient to impart what appears to be a smooth, continuous flow. Although somewhat obvious, it is important to point out for a couple of key reasons: (i) given that the frame rate of the still pictures is finite, it is obvious that information is lost in the capture process, and (ii) in any still picture there is bound to be a great deal of repetitive and redundant information.

The basic trick then in the compression of video (whether it originates from film, television camera, or server is immaterial) is to take these two premises to their ultimate conclusion. That is, reduce the information transmitted to the point whereby the image quality is acceptable, remember that "what is acceptable" will be governed by the particular application at hand.

It is not possible to provide a complete review of all of the various compression types within this paper, however, there are many articles which have been written on the subject and we encourage you to do further reading on the subject<sup>1</sup>.

Compression formats can be summarized into four types as follows:

- JPEG
- Motion JPEG (or M-JPEG)
- MPEG-1
- MPEG-2

The computer industry has been using lossless data compression techniques for many years in order to cram more data onto hard disks and for modem transmission. In fact, it was the computer industry which came up with JPEG (Joint Photographic Experts Group) standard for compressing high-resolution digital still pictures. The progression to M-JPEG was fairly rapid as the novelty of moving video on the computer was too tempting to resist.

The Moving Picture Experts Group (MPEG) was formed in 1988 in order to formulate international standards for the digital compression of moving pictures, particularly to satisfy the growing interest in CD-ROMs.

M-JPEG and MPEG-1 techniques are now used extensively for computer imaging and can be very cost-effective for disk recording, CD-ROMs, etc. but neither offer the optimum results required for broadcast level quality. What has emerged, as a result, is the MPEG-2 standard (an ISO/IEC ratified standard).

The following is a summary of the various compression formats indicating their respective bandwidths, uses and restrictions where applicable.

### JPEG

Basically, JPEG is used only for single still images. Quality can be very good as the processing is almost always non-real time and therefore the results can be optimized. Also, it is relatively easy to control the bandwidth, again due to the non-real time nature of the process.

---

<sup>1</sup> Recommended - The SMPTE Journal, Vol. 105, No. 2, February 1996.

## M-JPEG

M-JPEG relies on the processing techniques of JPEG in order to accommodate a set of still images on a field by field basis. The bandwidth becomes much more of an issue as we are now dealing with real-time processing. Unless great care is taken in the implementation, the bandwidth can quickly become unmanageable. An optimal solution is use a two-pass compression process whereby the results of the first pass are used to optimize the second pass, thus producing better results with tighter control of the bandwidth (more on this process later).

## MPEG-1

As previously mentioned MPEG-1 is used primarily with CD-ROMs. While it can be used for broadcast applications, the results generally negate any benefits. The bandwidth utilized is typically 1.25Mb/s and the quality (a subjective issue at best) can be described as "VHS like". MPEG-1 utilizes a 4:1:0 video format. What this means is that there is only one chrominance sample for every four luminance samples.

As a side note, full broadcast quality video is 4:2:2. That is, for every four luminance samples there are two samples each of the color difference signals, Cr and Cb.

## MPEG-2

The MPEG-2 standard has been described as more of a "tool kit" than a standard and perhaps this is fairly accurate. A standard generally defines a specific format, but the MPEG-2 standard provides a number of different modes of operation. Again, it is not possible to explore MPEG-2 in full detail within this paper, so we will focus on a couple of the more pertinent operating "modes"<sup>2</sup>.

The MPEG-2 tool kit consists of a number of operating levels and profiles. Levels refer to the available compression rates which may be employed for computer compressed data high definition television, at rates of less than 4Mb/s up to 80Mb/s respectively. Profiles refer to the compression type used which may vary from the full 4:2:2 signal to the elimination of complete frames.

The most common format used is the Main Profile at Main Level (MP@ML). This format utilizes a 4:2:0 structure. The quality associated with MP@ML is quite sufficient for distribution but will not survive in a production environment where multiple passes are required. One of the problems of MP@ML is the fact that GOP (Group Of Pictures) does not facilitate easy editing (due to the information content of the P and B frames).

---

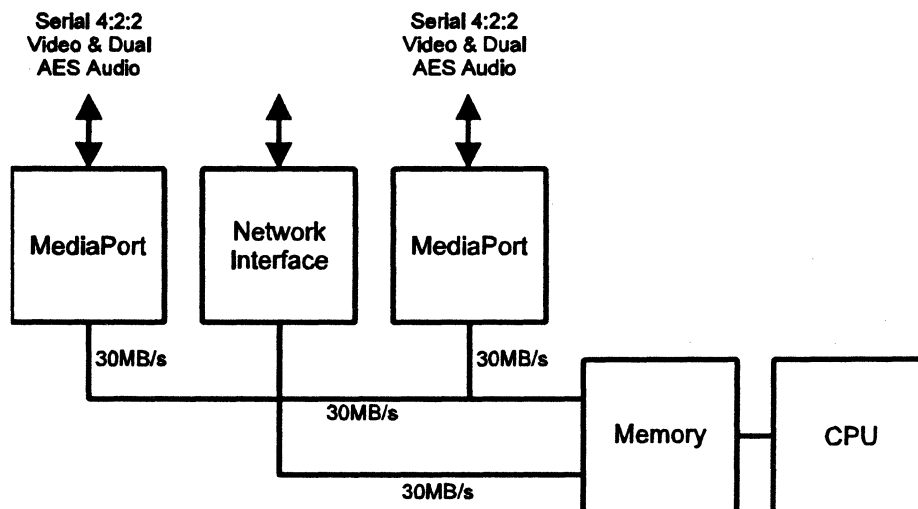
<sup>2</sup> Refer to Appendix A for an overview of the MPEG-2 standard.

The other format worth noting is the ML@4:2:2. The obvious advantage is that the number of chrominance samples, or blocks, is doubled from the MP@ML format, providing better resolution.

## Bandwidth Issues

Broadcast quality 4:2:2 video at 10-bits has an associated data rate of 270Mb/s, or 27MB/s. With the addition of two channels of AES audio (for a total of four monaural audio channels), the bandwidth requirement approaches 30MB/s.

The PCI Bus has a theoretical bandwidth of 132MB/s. As depicted in the typical system shown below, this means that the PCI Bus has the bandwidth required to support full broadcast video and audio.



Such a system would have uses as a commercial insertion file server for example, or possibly for editing applications. Note that the higher bandwidth requirements are reserved for the operation's revenue generation - commercials.

At M-JPEG compression ratios of up to 5:1 there are no visible artifacts and the quality is excellent. With ratios approaching 8:1 a few artifacts are visible but the quality is still very good.

## The PCI Bus Interface Module - MediaPort®

The Leitch MediaPort module utilizes M-JPEG technology. The reason for this is that the achievable quality is very good and the hardware is both readily available and meets physical and practical design considerations.

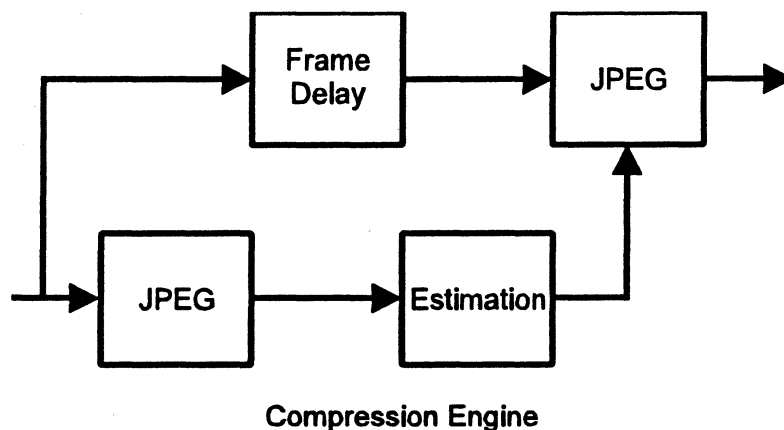
The MediaPort interface module has the following attributes:

- support of full 10-bit 4:2:2 digital video I/O
- compressed and uncompressed video support
- dual AES I/O channels
- timecode I/O
- full genlock
- audio and VBI remain uncompressed

Leitch designed the MediaPort interface module in conjunction with Digital Equipment Corporation. It was determined that one of the key features that the module should have was the ability to handle full bandwidth, uncompressed video. Please refer to the block diagram of the MediaPort which follows.

Another key element in the overall design was the handling of the associate audio channels and the vertical blanking interval (VBI). Although AES audio can be compressed there was really nothing to be gained in this particular case and therefore it was decided to leave it uncompressed. It was important to preserve the VBI also, so that user data, such as Closed Captioning, would not be destroyed. Therefore the VBI is left uncompressed (lines 14 to 21) regardless of the compression used for the active video portion of the signal.

In uncompressed mode, the video data is buffered and then routed directly to the PCI Bus, or is routed from the PCI Bus to the output. The module also supports an E-E mode of operation (electronics to electronics) whereby the input video is passed, via the buffers, to the output.



The compression engine shown on the previous page is the heart of the MediaPort module. Standard M-JPEG compression techniques will generally suffer from one of two problems: Either the resulting image quality is highly variable, or the size of the resulting compressed image will be highly variable.

Neither of these two problems were acceptable in our design. Therefore it was determined that the optimum method of performing the compression was to utilize a two-pass system as depicted in the aforementioned drawing.

Very simply put, the image is compressed twice. The first compression pass is used to modify the algorithms used in the second pass (the video data is delayed by an amount corresponding to the JPEG compression pass). This provides a much superior result in terms of image quality and also allows the size to be controlled (which translates to a fixed compression ratio).

### **Design Challenges**

As with any "first time" venture into a new area, there is an associated learning curve. Surprisingly, the implementation issues associated with the PCI Bus itself proved to be the least of our problems.

Since one of our main requirements was to support full bandwidth, uncompressed video, we truly required a bus bandwidth of 60MB/s. This was due to the fact that the data was transferred into buffers in memory and then subsequently into the storage media. The early available hardware showed itself to be somewhat challenged by this requirement to say the least. The PCI to host bridges were inadequate in terms of bandwidth performance.

One remaining problem was the size of the on-board DMA FIFO buffers used within the interface chip. In order to ensure the maximum data transfer possible it was necessary to spend considerable time arbitrating the "fill" or "empty" the buffer actions (depending upon whether the action was "record" or "play"). Again, this was due to the real-time nature of the operations involved and the quantity of data which had to be handled.

### **Conclusion**

Happily many of the above issues have since been resolved and PCI Bus implementation of real-time broadcast video has been eased to the point where it is possible to utilize a Pentium platform using an NT operating system.

## Appendix A

### An Overview of the MPEG-2 Standard

#### JPEG, MPEG-1 and MPEG-2

The computer industry has been using lossless compression techniques for many years to cram data onto hard disks and for transmission over modems. It was the computer industry that came up with the JPEG (Joint Photographic Expert Group) standard for compressing high resolution digital still pictures and it wasn't long before somebody thought it would be 'cool' to show video on their computer too, so along came *motion* JPEG. The MPEG (Moving Picture Expert Group) was formed in 1988 to determine international standards for the digital compression of moving pictures, particularly to satisfy the growing interest in CD-ROM's.

Motion JPEG and MPEG-1 techniques are now used extensively for computer imaging and can be very cost effective for disk recording, CD-ROM's, etc. but neither offer optimum results for broadcasting. What has emerged however is MPEG-2 (an ISO/IEC ratified standard) and the industry has adopted this at an amazing speed, driven almost entirely by the strong desire to provide viewers with a huge choice of programs delivered direct to home (DTH) via satellite or cable TV, using set top decoders.

#### The Technical Challenge

As the purist will testify, you cannot compress video to any extent without throwing something away and thereby reducing the picture quality. Fortunately however, the human visual system is incapable of absorbing all of the material presented in a complex moving image, so by skilfully choosing compression techniques which selectively discards information which the eye is unlikely to notice, impressive results can be achieved.

Each television picture comprises a finite number of tiny *pixels*. In the conventional 4:2:2 representation of NTSC and PAL television, there are 720 pixels along the active part of each horizontal line. In NTSC there are 486 active lines per frame (576 active lines in PAL) and 30 frames per second (25 in PAL). Each pixel is made up of 8 bits for luminance and 4 bits each for the two color difference signals (R-Y and B-Y, also known as  $C_r$  and  $C_b$ ), a total of 16 bits. So the bit rate for the active part of the video only, is:

NTSC	$720 \times 486 \times 29.97 \times 16 \sim 168\text{Mbits/s}$
PAL	$720 \times 576 \times 25 \times 16 \sim 166\text{Mbits/s}$



The sole purpose of MPEG-2 is to reduce these bit rates to something more manageable and its success relies on data reduction primarily in two areas of the motion picture. The first area is the information contained in each frame (spatial; relating to space, e.g. surplus blue sky, etc.) and the second is detail which does not change from frame to frame (temporal; relating to time).

## Levels and Profiles

Much credit must go to the MPEG team for the international standardization of MPEG-2. The published ISO/IEC documents 13818 (-1 to -4) cover video and audio compression and the multiplexing structure needed for combining video, audio and timing data for successful reproduction of video with synchronized audio. Not only is MPEG-2 truly a world standard, but the system encompasses everything from computer compressed data rates of less than 4Mb/s, through conventional TV at 10 to 15Mb/s and High Definition Television operating at up to 80 Mb/s. These are known as different *levels* and the MPEG-2 architecture supports all the levels shown in Fig. 1.

		PROFILES					
		Spatial resolution layer	Simple	Main	SNR	Spatial	High
<b>LEVELS</b>	<b>High</b>	Enhancement		1920 x 1152 60			1920 x 1152 60
		Lower		-			960 x 576 30
	<b>High-1440</b>	Enhancement		1440 x 1152 60		1440 x 1152 60	1440 x 1152 60
		Lower		-		720 x 576 30	720 x 576 30
	<b>Main</b>	Enhancement	720 x 576 30	720 x 576 30	720 x 576 30		720 x 576 30
		Lower	-	-	-		352 x 288 30
	<b>Low</b>	Enhancement		352 x 288 30	352 x 288 30		
		Lower		-	-		

Notes: 1920 x 1152 represents samples per line x lines per frame  
60/30 represents frames per second.  
The enclosed box represents conventional television (MP@ML)

Level	Low	Main	High-1440	High
Mb/s	4	15	60	80

Levels and bit rates - main profile

Fig 1 Profiles and levels

MPEG-2 also provides for flexibility in the type of compression used for each level. Compression types are known as *profiles* and may vary from use of the full 4:2:2 signal at the *high* end, to the elimination of complete frames at the *simple* end. Encoders can vary considerably depending upon the application, so details of the encoding scheme must be transmitted along with the data, to enable the decoder to reconstruct the signal. In this way encoders can be designed to handle the various levels using different *profiles* at the same time as keeping the cost of the decoders to a minimum for the desired application. Most 525 and 625 line broadcasting uses *main profile at main level (MP@ML)*.

## Layers and Scalability

One of the most ingenious features of MPEG-2, is its ability to transmit video signals of widely ranging quality. A relatively inexpensive MPEG-2 decoder can reconstruct a useful picture by using only part of the encoded video *bitstream*, the rest of the data being reserved for quality enhancements. Coded video data consists of a series of video bitstreams called *layers*. The first layer is known as the *base layer* and this can always be decoded independently. The other layers are called *enhancement layers*.

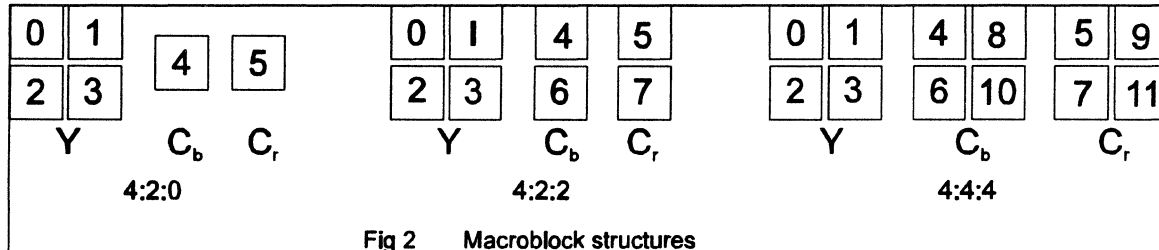
These layers may be used for spatial, temporal and other *scalable* extensions. (More information on this in next month's article on HDTV). If there is only one layer, the coded video data is said to have a *non-scalable* video bitstream. If there are two or more layers, the data is said to have a *scalable hierarchy*. Scalability has a further benefit, in that it helps to make the video resilient to transmission path errors. Transmission paths with the best error performance can be reserved for critical base layer information, while the enhancement layer data can be sent over a channel with inferior error performance.

## Video Bitstream

The video bitstream is made up of blocks of pixels, macroblocks, pictures, groups of pictures and video sequences as follows:

- Block
- Macroblock (MB)
- Slice
- Picture
- Group of Pictures (GOP)
- Video Sequence

The smallest element, a *block* consists of 8 lines x 8 pixels per line. Blocks are grouped into *macroblocks (MB)*, according to one of the MPEG-2 predefined *profiles*. The 4:2:0 macroblock format has 4 blocks for luminance, 1 block for  $C_r$  and 1 block for  $C_b$ . The 4:2:2 MB format has 4 luminance blocks, 2  $C_r$  blocks and 2  $C_b$  blocks. 4:4:4 again has 4 luminance blocks but this version contains 4  $C_r$  blocks and 4  $C_b$  blocks. These are illustrated in Fig.2. As can be seen, a 4:2:2 MB will contain 8 blocks and therefore 8 x 8 x 8 (512) pixels.

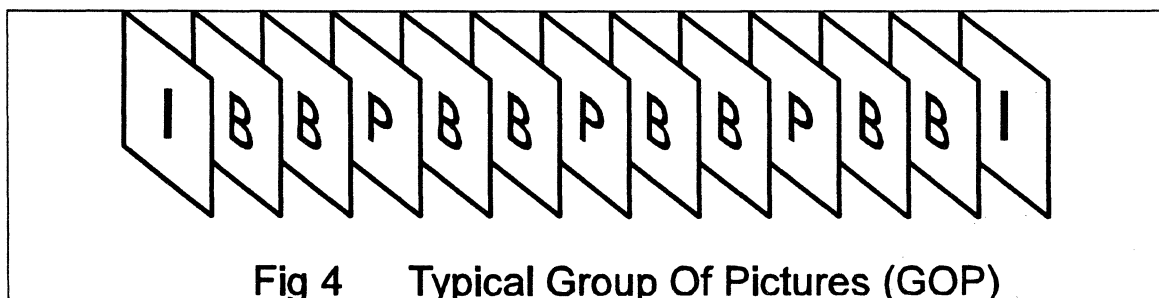


*Slices* are strings of macroblocks arranged horizontally along the raster. Slices can vary in length from a minimum of one macroblock to a maximum of one line. Pictures and groups of pictures will be examined during our discussions about temporal compression.

### Temporal Compression

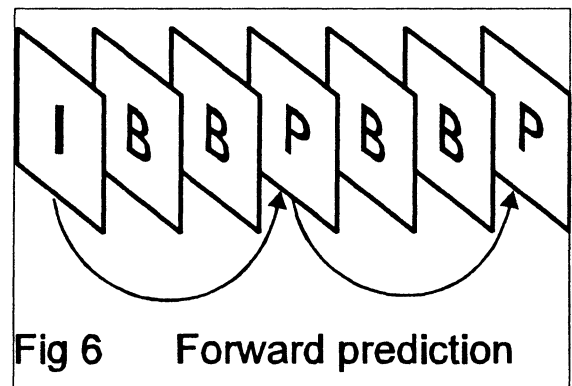
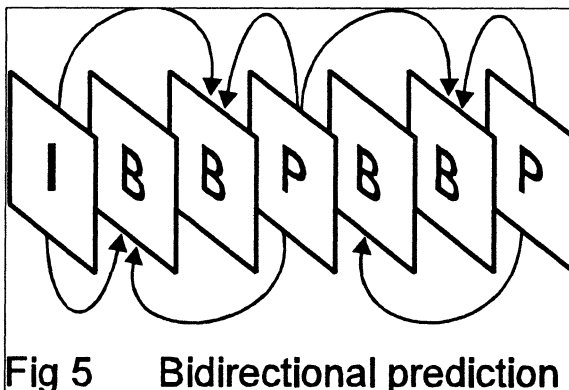
Temporal compression is designed to minimize the duplication of data contained in successive pictures. This is achieved by only transmitting motion vector data and not the whole picture over again. To facilitate motion predicting, MPEG-2 separates the video into 3 types of pictures (see Fig 4):

- I (Intra-coded) Pictures
- P (Predictive coded) Pictures
- B (Bidirectionally interpolated) Pictures



*I-Pictures* are the key reference for the other two picture types. They are derived by compressing the information in a single chosen field or frame (spatial compression). Still pictures are best preserved by using complete frames, but as the field rate is 2 x the frame rate, movement is better served by using field based pictures. Some MPEG-2 encoders are capable of analyzing the incoming video to determine the changes between successive fields. If there are no changes between odd and even fields, the encoder presumes that the two fields are part of the same frame and encodes them as such.

Changes between fields are noted and converted into motion vectors, which are encoded into data for later interception by the decoder. In this way, substantial bit rate reduction is achieved. The changes are transmitted in the form of *P-Pictures* and *B-Pictures*. *P-Pictures* are *predicted* directly from the previous *I-Picture* (see Fig 5). *B-Pictures* are derived using either *I-Picture* or *P-Picture* information and these reference sources may be either ahead of or behind the *B-Picture* being created (see Fig. 6). Hence the term *bi-directional interpolation*. Both *P* and *B* type pictures are also compressed spatially prior to transmission. The technique of motion compensation using the above method is known as *temporal* compression.



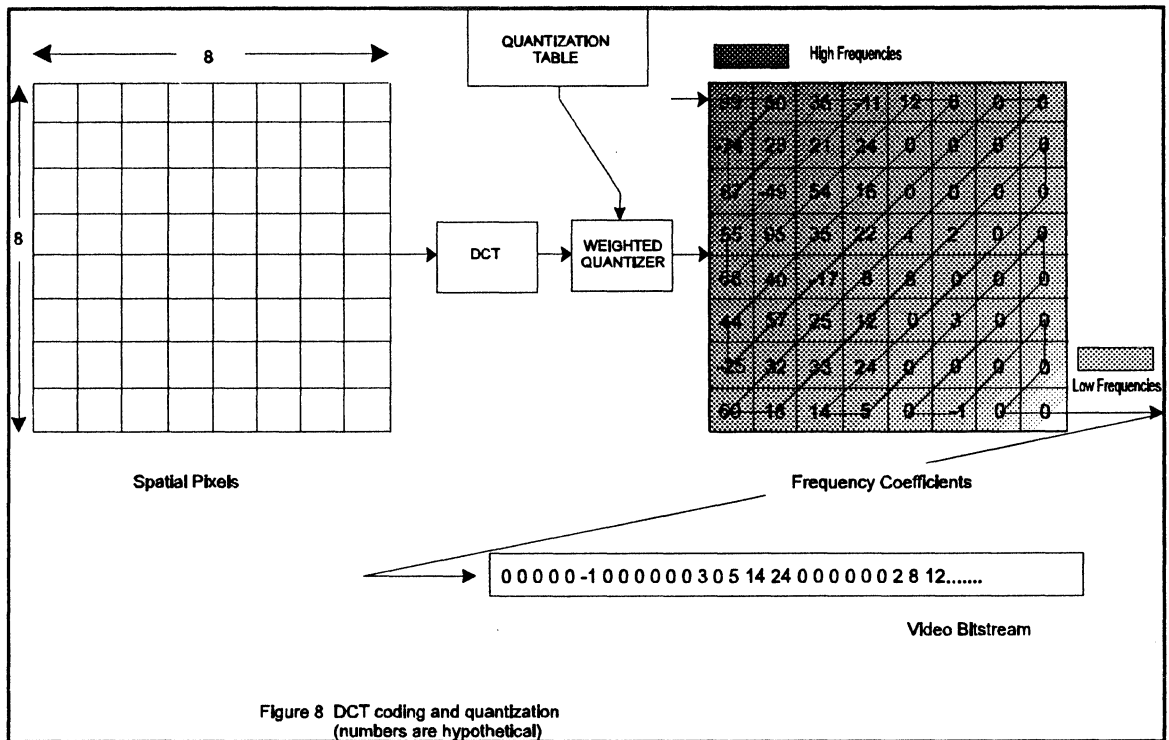
The three types of pictures are transmitted sequentially in a *Group of Pictures (GOP)* as shown in Fig 4, with the first picture always being an *I-Picture*. There are typically 12 pictures in a *GOP*, but as stated before, some encoders can detect changes between successive fields and, if the change is substantial, the encoder assumes that there has been a scene change, so it forces a new *I-Picture*. This causes the sequence to start over again. The *GOP's* are sent in a *Video Sequence*, which contains data defining picture size, rates and quantization matrices. The video sequence and all elements down to the slice size, provide unique start codes to facilitate detection by the decoder.

The only drawback of generating these *virtual pictures* is that engineers have yet to find an easy way to edit on B or P pictures. Consequently television stations are likely to continue using motion JPEG techniques for in-house television *contribution* (as you can edit on any field) , until a solution is found for this problem. Nevertheless compression ratios of up to 10:1 can still be achieved using JPEG. Compression ratios in the order of 25:1 are achievable with MPEG-2. MPEG-2 is considered to be a *distribution* compression format.

## **Spatial Compression**

The word *spatial* refers to the space in a single picture and the goal of spatial compression is to minimize the duplication of data in each picture. Bit rate reduction in spatial compression, is achieved by first transforming the video data from the time domain into the frequency domain using the *Discrete Cosine Transform (DCT)* method and then applying *quantization* and *variable length coding* techniques to reduce the bit rate.

Video is normally displayed on a time based device such as a waveform monitor rather than on a frequency based spectrum analyzer, but to accomplish data reduction, we must first transform the video data into the frequency domain. This is where DCT (a trigonometrical formula derived from Fourier analysis theory) is used to transform the data in each block of 8x8 pixels into blocks of 8x8 frequency coefficients. In the frequency domain, most of the high energy (and therefore most noticeable) picture elements are represented by low frequencies at the top left corner of the block and the less important details are revealed as higher frequencies towards the bottom right. (See Fig. 8). Note that at this stage we have not yet discarded any bits.



After DCT encoding, the data is subjected to a *quantization* process, weighted to reduce data in the high frequency areas, where the eye is less sensitive. We use more bits per pixel to quantize the important low frequency coefficients and less bits per pixel for the high frequency coefficients. The DC components are normally quantized at 10 bits, because if we employ coarser quantization of very low frequencies, the blocks themselves can start to become visible in the pictures. We have now achieved the first step in spatial bit rate reduction.

To create the video bitstream, the 64 frequency coefficients are scanned in a zig-zag fashion from top left to bottom right and, as can be seen from Fig.8, the high frequency areas are represented by strings of zeros. Further data reduction can now be achieved by transmitting only the number of zeros instead of the usual values of the coefficients.

The last stage in the spacial compression process employs *Variable Length Coding (VLC)*. The encoder can assign shorter code words for anticipated events and longer code words for unusual events. In this way, unexpected changes in the picture are given the highest priority. Also, where there is a high degree of *correlation* between one part of the picture and an adjacent part of the same picture, there is said to be *spatial redundancy*. The DC coefficients are encoded using VLC and the current block is compared with a predicted value from the previous block. This helps to ensure that adjacent blocks are restored with equal brightness so that the blocks themselves are not seen. JPEG and MPEG systems use the above methods of spatial compression for bit rate reduction.

### **Program Streams and Transport Streams**

So we have compressed the video. Now what? Before we can store or transmit the data, we have to multiplex the audio, video and system information together.

There are normally two audio/video multiplexers. One takes the video and audio *packetized elementary streams* and produces the *program stream* and the other uses the same data to generate the *transport stream*. Program streams are normally reserved for robust transmission paths where errors are unlikely to occur. The program stream data packets may be of different lengths and can contain a relatively large number of bytes. A transport stream on the other hand, can contain one or many programs with one or many independent time bases. Multiple TV channels can therefore be multiplexed together in this way. Transport stream packets are always 188 bytes in length. The transport stream is designed for use in environments where errors are likely to occur.

### **Conclusions**

JPEG and MPEG compression techniques will continue to be used for low cost computer compression requirements and wherever editing is needed in professional television.

MPEG-2 has become the international standard for video compression for any signals which are to be simply stored, distributed and viewed. CD-ROM's are being developed employing MPEG-2 compression methods. MPEG-2 has been adopted world wide as the compression standard for satellite delivered Direct To Home (DTH) television and for future cable and Digital Terrestrial Television (DTTV), including High Definition Television (HDTV).

## BOARD IMPROVES JPEG COMPRESSION USING PRE- AND POST-COMPRESSION IMAGE SCALING

Harold Schiefer, Ernest Yeung, Steven Hanna and Lance Greggain

Genesis Microchip Inc.

200 Town Centre Blvd., Suite 400

Markham, Ontario, Canada

(905)470-2742

E-mail: harold/ernest/steven/lance@genesis-microchip.on.ca

### ABSTRACT

This paper describes a PCI board that combines image scaling and JPEG compression in order to increase compression ratios without sacrificing image quality. By using high-quality scaling before compression, the controlled reduction in image bandwidth can reduce DCT coefficient quantization artifacts, which are visible as “blocks” in an image. The board demonstrates the processing of real-time video and still images. It uses the Genesis Microchip gm833x2 for image scaling and the C-Cube CL560 for JPEG compression. This paper discusses the board architecture, compression improvements with horizontal and vertical scaling for various scale factors and the implications for moving video and images on the PCI bus. While this paper describes the use of image scaling with JPEG compression, other compression methods also benefit from image scaling.

### INTRODUCTION TO IMAGE SCALING AND COMPRESSION

Bandwidth and storage limitations have always been the biggest obstacles to the popular use of digital imaging technology. Applications such as video editing, teleconferencing, video-on-demand, etc., all demand tremendous amount of storage space or transmission bandwidth. Image and video compression techniques offer a partial solution to the above problems - data is compressed, transmitted or stored, then decompressed when it is to be used. However, there is an upper limit to the achievable compression ratio - at very high compression ratios, details are lost and annoying artifacts are often introduced in the image.

Image scaling technology is an essential part of digital imaging. This technology is used from desktop “video-in-a-window” applications to the projection of huge images in conference halls. A correctly scaled image should be visually indistinguishable from the original - information lost should be minimized and no artifacts should be introduced. One important, although not obvious, application of scaling technology is in image compression. An image can first be shrunk before being fed into an image compression system, and the resulting image from the decompression process can be zoomed back to its original size. This additional step to traditional image compression can greatly reduce the amount of raw data to be compressed, thus enabling higher compression ratios to be achieved without sacrificing quality.

### Overview of Image Scaling

Although the problem of scaling a digital image has been around for some time, few satisfactory solutions exist. Popular scaling techniques such as pixel dropping and bilinear transformation offer simple solutions, but their results are often poor. Pixel dropping involves selecting the source pixel that is closest to the target pixel (see Figure 1, top). This technique is quick and simple, but the resulting images are often distorted and unwanted aliasing is often introduced. Linear interpolation generates a target pixel by taking a weighted average between the two closest source pixels. This technique offers slightly better quality than pixel dropping, but the results are still unsatisfactory and deteriorate quickly when the scaling factor is greater than 50% (see Figure 1, bottom).

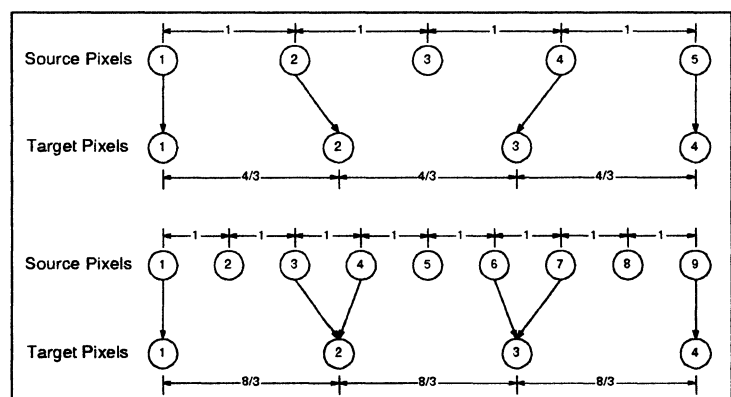


Figure 1: Pixel dropping (top) and bilinear interpolation (bottom)

There are two principal “correct” methods of scaling a digital image. The first one, which is extremely complicated, involves the use of switched filters. With this method, a target pixel is generated directly from a group of source pixels through



a filter. However, a new filter is required for each source and target image size combination, and for each pixel a new phase of the given filter is necessary. Thus, the number and complexity of filters required to implement this algorithm is high. Furthermore, in order to generate the exact phase shift required for each pixel, the filter coefficients must be precise. This method is seldom used.

The other “textbook-correct” method of image scaling involves the use of multirate DSP theory. The challenge in image scaling is changing the sampling rate of a signal. In order to properly scale an image, an intermediate stage is required in the process. The source pixels are interpolated in order to generate a large number of intermediate pixels, consequently increasing the sampling rate of the original signal. The target pixels are then obtained by filtering the intermediate pixels with a decimation filter, thus decreasing the sampling rate to the required rate. There are a number of difficulties with implementing the above scaling method, including complexity of control circuitry, large memory requirements and differences in input/output sampling rates. The family of high-quality real-time image scaling integrated circuits (ICs) offered by Genesis Microchip Inc. (Markham, Ontario) have overcome these difficulties in implementation. The scaling ICs have been implemented with a “silicon efficient” patented algorithm based on the “textbook-correct” multirate DSP approach.

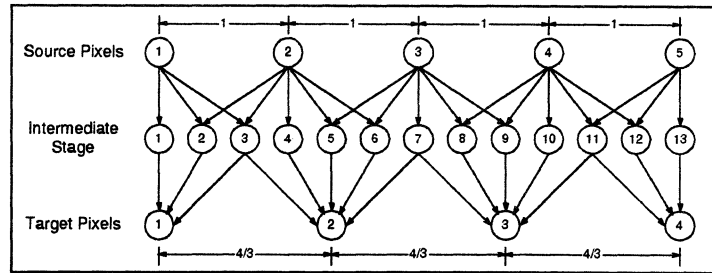


Figure 2: The "textbook-correct" multirate image scaling algorithm

### Overview of JPEG Compression

The JPEG (Joint Photographic Experts Group) standard has been in popular use in the imaging industry for the compression of natural images. JPEG implementations are usually lossy, which means that after the compression/decompression process, the resulting image is similar but not identical to the source. This compression algorithm divides an image into blocks of 8x8 pixels, and samples the pixels with cosine functions in order to obtain their frequency components. Since the human eye is insensitive to high frequencies, many of the high frequencies can be coarsely quantized (or reduced to 0) without adversely affecting the quality of the image. The remaining frequency components are then rearranged in a zigzag manner, and the Huffman encoding scheme is used to compress the data. In order to decompress an image, the above process can be reversed.

It should be noted that the JPEG compression algorithm (or any other compression algorithm, for that matter) cannot be used to infinitely compress an image. At compression ratios beyond a certain threshold, too many high-frequency quantization levels would have to be removed, resulting in missing details and blocky images. Thus, in order to achieve the extremely high compression ratios required in today’s imaging applications, image scaling can be used in association with JPEG compression.

### Benefits of Combining Image Scaling and Compression

There are a number of benefits in combining image scaling with compression. By shrinking an image before it is fed into the JPEG compression engine, the amount of raw data to be compressed is significantly reduced. (If an image is shrunk by 50% on each side, the amount of raw data is reduced by 75%.) Thus, the effective compression ratio achievable is greatly increased. Moreover, during the decimation stage of the scaling process, high-frequency components in the image are reduced. This reduction facilitates the quantization step of the JPEG compression process. It should be noted that although only JPEG compression is covered in this topic, other image compression algorithms also benefit from scaling. For example, MPEG (Moving Pictures Experts Group) compression can benefit in a similar way as JPEG, since the two are closely related. The emerging wavelet compression technique can also be paired with scaling, with an additional benefit. Since the optimal use of a filter bank requires a signal to be of length  $2^n$ , with n being an integer, an image scaler can be used to resize a rectangular image to a  $2^n \times 2^n$  square before it is fed into the wavelet compression engine. The result can be scaled back to its original shape.

In order to provide a reference design for developers and to demonstrate the benefits of pairing image scaling and compression, Genesis Microchip and C-Cube Microsystems have joined forces to produce PCI-833/560 “Edit Pro”, a PCI board which performs JPEG compression with scaling on still and video images.

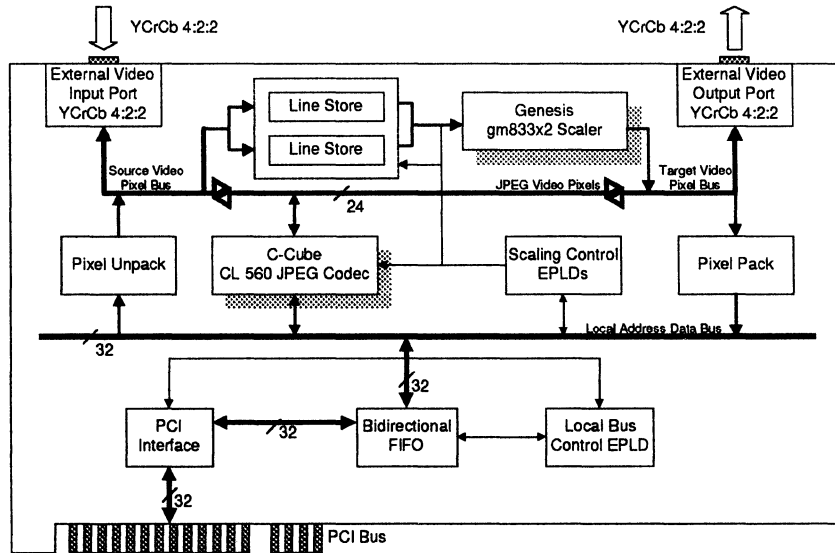


Figure 3: Block diagram for Edit Pro

### Data Processing Modes

The board architecture provides the processing and data flows typically used for video editing and image storage/retrieval operations. Four primary data processing paths provide support for the capture of external video, playback of previously captured video, as well as memory-to-memory compression and decompression operations. Additional paths support video input to output scaling and memory-to-memory scaling.

While in the External Video Capture mode, the Edit Pro accepts digital YCrCb data from an external video source and processes the video by first scaling and then JPEG compressing the video stream. The resulting compressed video stream is then fed to the PCI bus using bus master burst transfers to system memory, where the CPU manages the subsequent storage of compressed data onto the system hard disk. The scaled data prior to JPEG compression is also available through the digital video output port for monitoring the captured video in real time on an external display device (see Figure 4).

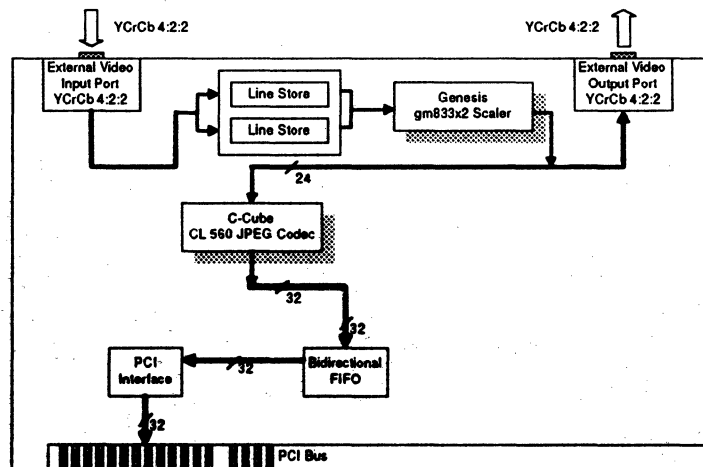
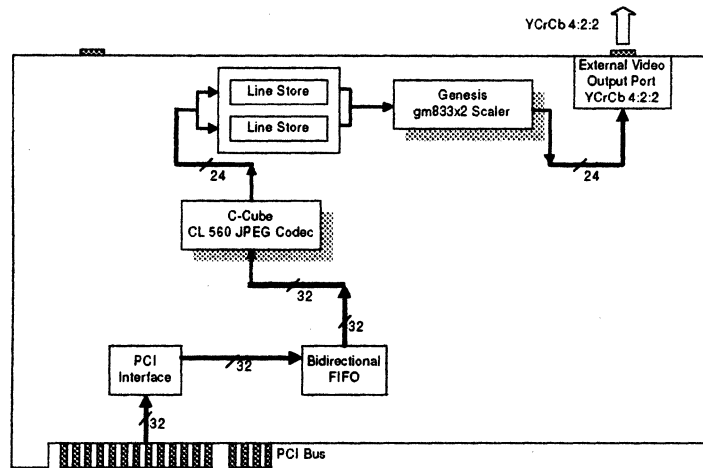


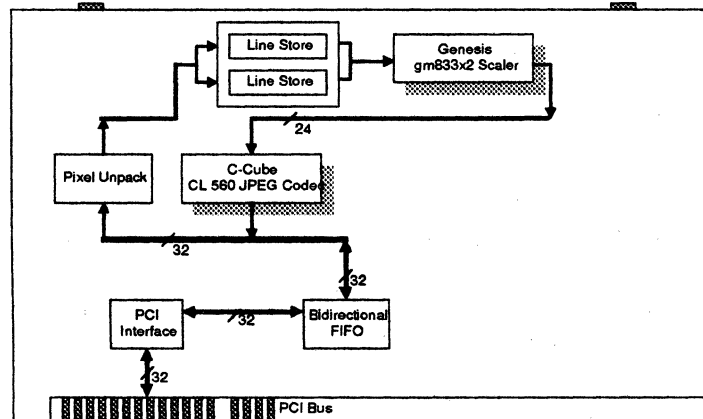
Figure 4: External Video Capture mode

In the External Video Playback mode, the Edit Pro performs PCI bus master burst transfers from system memory to the board's JPEG engine for decompression. The decompressed image is then scaled (zoomed or shrunk) with the results then fed to the External Video Output Port. The host CPU performs file management by accessing compressed video from the system hard disk and writing the retrieved compressed image data to the buffers in system memory in real time. This mode is typically used for playing back video clips that have been edited or rendered into compressed files on the system hard disk (see Figure 5).



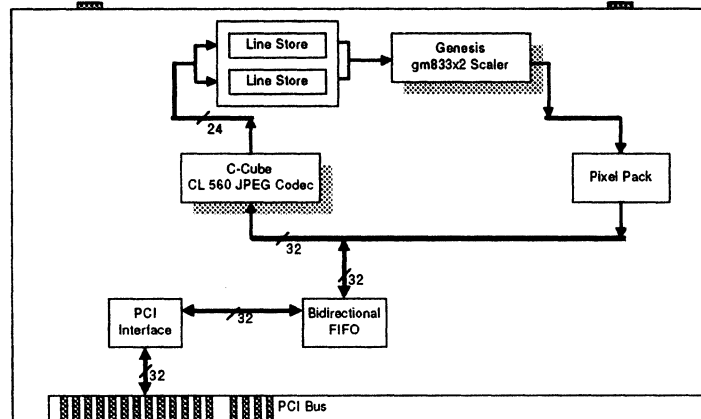
**Figure 5: External Video Playback mode**

In the Memory-to-Memory Compression mode the Edit Pro performs PCI bus master burst transfers from system memory to the scaler which pre-scales video data for the JPEG compression engine. The JPEG engine compresses the image data and feeds the results to the PCI interface which performs burst transfers to the system memory for subsequent storage on the system hard disk. Memory-to-memory compression is typically used for re-compressing fields or frames modified during an editing or rendering process (see Figure 6).



**Figure 6: Memory-to-Memory Compression mode**

While in the Memory-to-Memory Decompression mode, the Edit Pro transfers compressed fields or frames using PCI bus master burst transfers from system memory to the board's JPEG engine. The resulting uncompressed data stream is routed through the scaler data path and then fed to the PCI bus using bus master transfers. The decompressed and scaled data is transferred to system memory for subsequent storage on the hard disk. This mode is typically used for expanding compressed fields or frames so editing functions can be performed (see Figure 7).



**Figure 7: Memory-to-Memory Decompression mode**

### PCI Interface

The Edit Pro's PCI interface consists of an AMCC PCI interface controller and bi-directional synchronous FIFO memory. The PCI controller provides support for bus master DMA operations and is capable of handling concurrent bus master read and write flows to and from the PCI bus. These simultaneous burst transfers can occur independently of the host processor.

The FIFO memory, or "Up/Down FIFO", is provided to compensate for the latency in acquiring the PCI bus. Once the bus is acquired, data can be bursted to or from the PCI bus at a peak rate of 132 Mbytes/sec. The Up/Down FIFO also provides a mechanism for crossing from the PCI clock domain to the CODEC clock domain.

Also provided with the PCI interface are mailboxes. All on board registers are accessed through the mailboxes which provide a bi-directional data path for transmitting control/status information to and from the board. These mailboxes are accessed through the PCI interface and can be either I/O mapped or memory mapped.

The Edit Pro provides support for expansion BIOS code. This code would share space in a serial EEPROM with a boot-up configuration image for the PCI controller. It would allow the system designer to establish field upgrade strategies. The PCI specification establishes a standardized interface between software and hardware peripherals via this BIOS. The expansion BIOS ROM is limited to a 2K byte serial NVRAM, of which a portion would be dedicated to the initialization of the PCI controller.

### LAD Bus and JPEG CODEC Interface

The Local Address Data (LAD) bus connects to the host bus interface of the CL560 and provides both compressed video data, decompressed data, and CL560 initialization and control. The host bus HBUS[31:0] is a multiplexed address and data bus. For video data transfer, the CL560 provides an interface to the CODEC FIFO that eliminates the address phase on the data bus.

For video capture and compression, the CL560 is programmed to generate a DMA request (DRQ) when its CODEC FIFO is not empty. The DRQ signal indicates to the LAD controller that data is available in the CODEC FIFO and data will be transferred over the LAD bus to the Up/Down FIFO. At the end of a file, the FRMEND interrupt in the CL560 is triggered when the last word of the compressed field is processed by the CL560 and subsequently transferred to the Up/Down FIFO. Depending on PCI bus latency, data is either accumulating in the Up/Down FIFO or being bursted to memory over the PCI bus.

For video decompression and playback, the host processor configures the controller for PCI bus mastering. This allows the controller to independently move data from system memory into the Up/Down FIFO. The LAD controller then transfers data over the LAD bus and into the decompression engine of the CL560. The CL560 is programmed to generate a DRQ output whenever its CODEC FIFO is not full. The DRQ indicates to the LAD controller that a word can be transferred to the CODEC FIFO. Data is transferred to the CODEC FIFO until the end of the field has been reached.

The LAD bus also connects indirectly to the Source Video Pixel Bus (SVPB) and Target Video Pixel Bus (TVPB) (See Figure 3). If scaling is required prior to compression, raw pixel data is routed from the PCI bus to the SVPB for pre-

compression scaling. If video scaling is required after decompression, data is routed from the TVPB onto the LAD bus, through the Up/Down FIFO, and finally out onto the PCI bus. The LAD bus connects to the host interface of other devices on the board and is used for updating control parameters in the gm833x2 scaler and the various control EPLDs. Therefore the LAD bus also provides a mechanism to transfer status and control information to and from these devices.

## **External Video Input and Output**

External digital video is fed to the Edit Pro board using a 34-pin ribbon cable connector that supplies a 2x pixel clock, pixel controls such as HSYNC, VSYNC, BLANK, PIXEL\_ENABLE and the 16-bit YUV 4:2:2 pixel stream. An external video I/O board can be used to decode analog video inputs (NTSC/PAL, SVHS/Composite) to generate the digital video stream used as input to the Edit Pro. Alternatively, the design could be modified to provide on-board analog video decoding to generate the digital video input stream.

The output process is similar. Decompressed, scaled or pass-through video output is available at another 34-pin ribbon cable connector that supplies a 2x pixel clock, pixel controls such as VSYNC, PIXEL\_ENABLE as well as the 16-bit YUV 4:2:2 pixel stream. This digital video output is used to transfer pixels to an external video I/O board that provides encoding to generate analog video outputs (NTSC/PAL, SVHS/Composite) for an external display device. Alternatively the design could be modified to provide on board analog video encoding to generate the analog video output.

A video I/O board (PC833x2-T) is available from Genesis for use with the Edit Pro board to provide the analog video input/output capability. The Edit Pro design is modular so systems integrators with other video I/O requirements can use the existing I/O digital video connectors and tailor the EPLDs to meet specific video I/O requirements.

## **Image Scaler Datapath**

The image scaler datapath consists of a zoom buffer, the Genesis gm833x2 scaling engine, an output FIFO, and EPLD controllers. The zoom buffer contains two dual-port SRAM devices, one for the Y data and one for UV (CrCb) data. Source video data is fed to the scaler path from the external video input port, PCI bus interface (via the LAD bus), or the JPEG engine. Lines of source video data to be scaled are written into the zoom buffer, which can store two lines of 1024 pixels per line. When a source line has been written into the zoom buffer, the zoom buffer control EPLD then begins transferring that line into the gm833x2 scaler for scaling. While the current line is being scaled, the next source pixel line is captured into the other line store of the zoom buffer. The zoom buffer is continuously operated in this swap buffer arrangement until the entire image has been processed. The scaler runs at 27 MHz, which is twice the input pixel rate, so that during zoom operations more output pixels than input pixels can be generated during one input source line period. The zoom buffer also provides the mechanism for crossing from the source video clock domain to the scaler clock domain. Also, the zoom buffer can repeat source lines as requested by the scaler during vertical zooming of the source image.

Cropping of the source image is implemented using the zoom buffer control EPLD to perform horizontal cropping of source lines by controlling addressing into the zoom buffer. Vertical cropping is supported using the internal vertical blanking and source image size parameters within the gm833x2 scaler.

The gm833x2 performs source image scaling using interpolation followed by low-pass filtering and decimation to generate the desired output image size. Up to 33 taps of FIR filtering are provided in both horizontal and vertical dimensions for optimal scaling quality within a single monolithic device.

An output FIFO is used after the scaling engine to buffer data for transfers over the Target Video Pixel Bus (TVPB). The TPB provides a path to the external video output port, the JPEG engine video pixel port, or the Up/Down FIFO used for queuing up burst transfers for the PCI bus.

## **PERFORMANCE RESULTS**

### **Quality Improvements on Still Images**

By adjusting scaling ratio and JPEG quality factor, a compressed image file can be generated with the same size as another file generated by JPEG compression alone. The compressed file generated with scaling will exhibit less JPEG DCT quantization artifacts, as the scaler filtering removes high-frequency components as part of the scaling process. Due to the reduction of the raw amount of image pixels, there is also less data for the JPEG engine to compress. Also, the JPEG engine

can be adjusted for finer quantization to reduce the JPEG DCT quantization artifacts. Effectively, the JPEG DCT quantization artifacts can be exchanged for the scaler's reduction in higher frequency components - in other words, a highly compressed JPEG image can be made less "blocky" if a slight increase in softness can be tolerated.

A still image of "Lena" was compressed with the Edit Pro board to demonstrate the reduction in DCT quantization artifacts when scaling is used. For each image that was compressed with JPEG only, a corresponding hybrid compressed image was generated using a mix of compression and scaling. The hybrid compression had the scaling and Q factors adjusted to achieve the same target compressed file size as the JPEG-only compressed file. With similar file sizes, a quality comparison between the JPEG- only and hybrid compression methods can then be made.

The measurement of reduction in DCT quantization artifacts achieved by the hybrid approach is subjective and must be evaluated by viewing images. The raw full-size image and resulting file sizes for various Q factors and scaled sizes are shown in the following table. (The raw full-image size was 720 pixels by 480 lines.)

<i>Test</i>	<i>Q Factor</i>	<i>Pre-compress scaling shrink frame format</i>	<i>Post-compress scaling zoomed frame format</i>	<i>File size</i>	<i>File size - percent original size</i>	<i>Artifacts</i>
Raw	- N/A -	- N/A -	- N/A -	691,212	100.0	None, full-size unprocessed image
JPG11	20	No scaling	720x480	54,276	7.9	No artifacts
JSC11	15	624x480	720x480	54,956	8.0	No artifacts
JPG4	85	No scaling	720x480	21,452	3.1	Some DCT artifacts
JSC4	22	304x408	720x480	21,536	3.1	No artifacts
JPG3	150	No scaling	720x480	15,524	2.2	Extreme DCT artifacts i.e., "blocks" clearly visible
JSC3	23	208x400	720x480	15,380	2.2	No blockiness, image slightly softer

The images JPG3 and JSC3 have been reproduced on the next page.



Figure 8: Image JPG3 - JPEG only compression



Figure 9: Image JSC3 - JPEG compression with scaling

## Reduced File Sizes on Video Sequences

By using high quality scaling before compression, the image size is reduced and therefore the amount of raw image data to be compressed is also reduced. When compressing fields of video with a constant JPEG quality factor of 20, compressed image file sizes could be varied linearly with the amount of scaling.

A video sequence titled "FLOWERS" was compressed by the Edit Pro board using JPEG-only compression during one capture sequence and using JPEG compression with scaling during another. The video sequence consisted of a video panning shot with numerous detailed and moving objects. In the background are trees, a house with many geometric lines, a windmill in which the windmill sails are rotating and a large quantity of flowers with petals moving in the breeze. The use of scaling together with compression resulted in significant reduction in file sizes, as well as reduced data throughput requirements during capture and playback. During playback, the motion video scaling resulted in minimal image degradation and some reduction in the amount of JPEG DCT quantization artifacts. The resulting data is summarized in the following table:

<i>Test</i>	<i>Percent original size</i>	<i>Q Factor</i>	<i>Pre-compress scaling shrink frame format</i>	<i>Post-compress scaling zoomed frame format</i>	<i>5 second video clip file size (bytes)</i>	<i>Artifacts</i>
Raw	100.0	- N/A -	No scaling (720x480)	720x480	103,680,000	None
JPGVID	12.9	20	No scaling (720x480)	720x480	13,360,004	Some DCT artifacts
JSCVID	6.8	20	Horizontal shrink (352x480)	720x480	7,078,088	Some DCT artifacts, softened image

## CONCLUSIONS:

Scaling can be used to enhance compression to achieve better compression ratios. At the same time, scaling can also reduce DCT quantization artifacts or "block" artifacts by trading image spatial bandwidth for less DCT quantization artifacts. The Edit Pro design provides the ability to vary both the amount of scaling (image size format) and the DCT quantization (Q factor). This results in an additional degree of freedom in controlling the compressed image file sizes, image bandwidth and DCT quantization artifacts as a result of the compression process. The use of scaling to enhance compression is also applicable to other DCT-based compression schemes such as MPEG, as well as other non-DCT-based algorithms.

Scaling is almost always in the video processing chain for playback applications such as picture-in-picture or overlay windows. By providing quality scaling in the video processing chain, additional benefits are provided to the compression/decompression process. Only good-quality scaling using multirate DSP techniques with proper FIR filtering can be used to enhance compression. Other scaling methods such as pixel/line replication/dropping or bilinear interpolation do not filter the image properly and may result in more high-frequency DCT quantizer coefficients. Poor scaling, therefore, reduces the efficiency of compression.

With this hybrid compression technique, file sizes are reduced and bandwidth demands on the PCI bus are also lowered. This provides opportunities for better live video throughput, more channels of live video, and ability to store more video data.

A possible future improvement to this system is the incorporation of image enhancement algorithms to improve the high frequency content of the decompressed images. This would result in sharper, higher quality images.

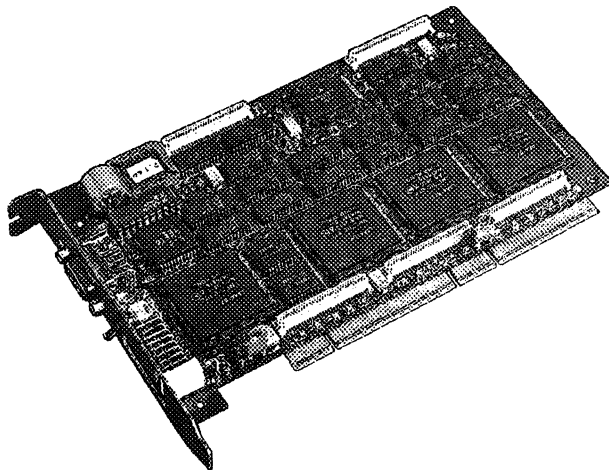


# PCI BUS ANALYZER SIMPLIFIES SYSTEMS TEST & DEBUGGING

Thomas Nygaard  
Vice President, VMETRO, Inc.  
1880 Dairy Ashford, #535, Houston, TX 77077, USA  
Tel.: (713) 584-0728, Fax: (713) 584-9034  
Email: thomas@vmetro.com

## **Abstract**

This paper discusses how VMETRO's new PCI and PMC analyzer boards can assist engineers who are involved in design, testing and debugging of PCI-based PCs, workstations or embedded computers. Operated from a PC or terminal, the analyzers are powerful tools that offers a considerable increase in debugging productivity, for hardware as well as software tasks, compared to general-purpose instruments.



*Figure 1. The PBT-315 PCI Bus Analyzer is a full-featured Logic Analyzer on a short PCI card form factor, for analysis of PCI systems like PCs, servers etc.*

## **Introduction**

Testing and debugging PCI bus systems can be a challenge. The comprehensive PCI specifications require careful timing design of the hardware, as well as a thorough understanding of a number of complex mechanisms for data transactions, error conditions and cache operations. In addition, software issues play a major role, when several complex devices must play together. Potential problems often relate to configuration

and initialization, hardware and software incompatibilities, incorrect byte-swapping, interrupts, and so on.

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 PCI Bus Analyzers that greatly simplifies test and debugging of PCI and PMC (PCI Mezzanine Card) systems.

## **Various Form Factors**

### **Short PCI card**

The model PBT-315 is an advanced self-contained Bus Analyzer for the PCI bus, implemented as a short PCI card. The analyzer is designed to be plugged directly into a spare slot on a PCI motherboard as found in PCs and servers, for immediate tracing capability of all PCI channels. This eliminates tedious installation and setup procedures as required by general purpose logic analyzers.

### **PMC - PCI Mezzanine Card**

Similarly, the PBTM-315 PMC Analyzer for PCI Mezzanine Cards plugs into a PMC slot on e.g. VMEbus boards. As a unique feature, the PMC version of the analyzer is equipped with both male and female PMC connectors, allowing one standard PMC module to be piggybacked on the analyzer, eliminating the need for a spare PMC slot. The analyzer may also operate from a separate power supply.

### **Compact PCI**

The Compact PCI standard, which is PCI with Eurocard mechanics for the embedded market, is supported by the PBTM-315 by means of a standard PMC adapter (carrier) card that fits in 3U or 6U sizes.

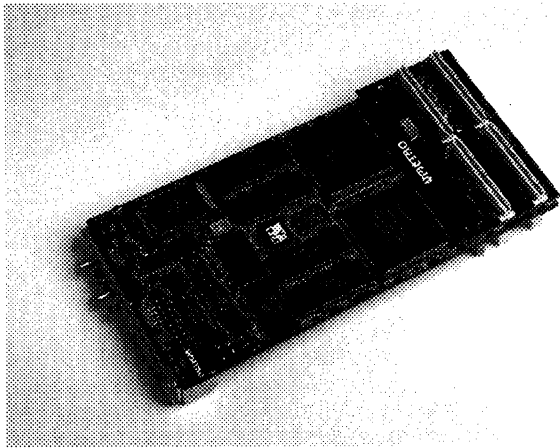


Figure 2. The PBT(M)-315 is a PCI Bus Analyzer for PCI on a PMC - PCI Mezzanine Card.

### Complex Triggering

To trigger on the most complex problems, VMETRO's PCI analyzers offer four parallel trigger words and a 16-levels trigger sequencer with If-Then-Else operators. The sequencer include 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 3 shows an example of the setup screen with a complex trigger condition.

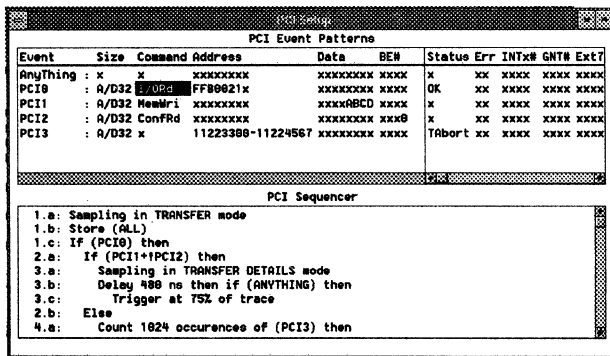


Figure 3 : PCI analyzer setup screen with example of how to form a complex trigger condition.

### Protocol-sensitive Bus Sampling

One of the most fundamental properties of the PBT(M)-315 is the protocol-sensitive sampling of bus cycles for state analysis. Unlike general-purpose logic analyzers, VMETRO bus analyzers know the bus protocol of the target bus to ensure that sampling takes place at the right moments. This gives a trace that clearly displays all kinds of bus activity, like arbitration, commands,

interrupts, cache cycles etc., and it ensures that the bus is not sampled at unimportant times.

### Demultiplexed Address / Data

The PCI bus multiplexes Address and Data into a common 32-bit bus. In a similar way, the bus COMMAND signals are multiplexed with the data byte enables (BE#). This saves system cost, since the number of pins on chips and connectors is reduced. However, it makes it more difficult to analyze the bus using a regular logic analyzer, since a given sample does not contain all information about a bus transfer.

To overcome this, the PBT(M)-315 has the capability to demultiplex Address/Data and COMMAND/BE# into separate trace channels. This is possible since the analyzer has as many as 128 trace channels, a luxury found only on the most expensive logic analyzers.

This important feature not only simplifies readability of the trace, but allows powerful triggers and store qualifiers involving both address and data to be defined easily.

Sample	Time	Latency	Size	Command	Address	Data	BE#	Status	Err	INTx#
TRIG	0ns			A/D32	MemRi	00000000	....55	....1110	OK	--
1	30ns			A/D32	MemRi	00000000	....AA	....1101	OK	--
2	120ns	60ns		A/D32	MemRi	00000000	FFFF	....0011	OK	--
3	150ns	60ns		A/D32	I/OrD	88888888	BBBBBBBB	....0000	OK	--
4	60ns			A/D32	I/OrD	88888888	DDDDDDDD	....0000	OK	--
5	60ns			A/D32	I/OrD	88888888	EEEEEEEE	....0000	OK	--
6	120ns	30ns		A64	MemRd	FEDCBA9876543210			OK	--
7	60ns			D32		11223344	....0000	OK	--	
8	60ns			D32		55667788	....0000	OK	--	
9	120ns	30ns		A64	MemInv	CAFECAFEABBA88BA			OK	--
10	60ns			D64		3333222211110000	00000000	OK	--	
11	60ns			D64		7777666655554444	00000000	OK	--	
12	30ns			A32	MRdLn	00100000			OK	--
13	60ns			A/D32	MRdLn	00100000	00112333	....0000	OK	--
14	30ns			A/D32	MRdLn	00100000	44556677	....0000	OK	--
15	60ns			A/D32	MRdLn	00100000	8899AABB	....0000	11001	PERR
16	60ns	30ns		A/D32	MRdLn	00100000	8899AABB	....0000	OK	--
17	270ns	180ns		A/D32	ConFRd	00200000	....1111		Abort	--
18	90ns	60ns		A/D32	MemRi	00400000	33333333	....0000	OK	--
19	30ns			A/D32	MemRi	00400000	44444444	....0000	OK	--
20	60ns			A/D32	MemRi	00400000	44444444	....0000	Abort	--
21	120ns	90ns		A/D32	MemRi	00300000	11111111	....0000	OK	--
22	300ns	210ns		A/D32	I/OrD	00000000	55555555	....0000	11001	SERR
23	210ns	120ns		A/D32	I/OrD	88888888	BBBBBBBB	....0000	OK	--

Figure 4 : The TRANSFER sampling mode gives a demultiplexed and decoded PCI trace, suitable for software analysis.

### Various Sampling Modes

When sampling the PCI bus, the PBT(M)-315 stores a 128-bit sampling word (full 64-bit PCI + 8 external signals + time tags and utility bits) into a Trace Memory 32K to 256K deep. To give the user the most suitable display for different applications, the analyzer can sample the activity of the PCI bus in four different ways.

The four Sampling Modes are:

**CLOCK sampling:** Stores one sample per each PCI CLK cycle. This captures all the details of how the PCI bus is exercised, clock-cycle by clock-cycle, useful to verify the behaviour of bus interface state machines etc. This mode is suitable for hardware analysis.

**TRANSFER sampling:** Stores one sample per valid Data Phase, each sample includes the Address and Command which is latched from the address phase. This is the optimum way to analyze bus transactions as seen from a software point of view.

**TRANSFER DETAILS sampling:** Stores one sample per each PCI CLK cycle only within a bus transaction, i.e. when the signals FRAME# or IRDY# are active. In this mode, all idle clock cycles are skipped, conserving space in the trace buffer.

**200MHz Timing Sampling:** Stores one sample every 5ns with an optional 200MHz Timing Analyzer piggyback module. This is for detailed HW analysis of PCI bus timing.

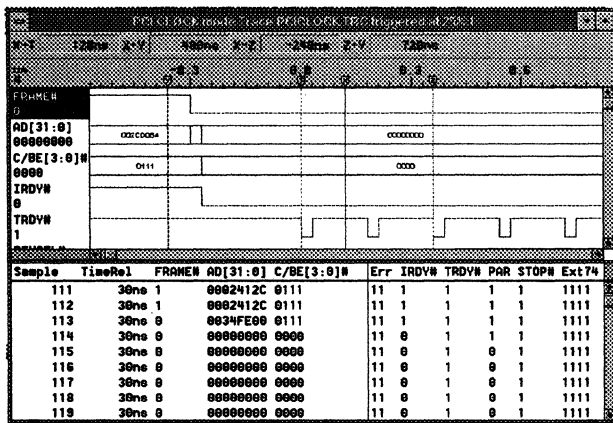


Figure 5 : The CLOCK sampling mode gives a raw undecoded PCI trace, suitable for hardware analysis.

### Slot-Specific or User-Defined Signals

The PCI bus has certain slot-specific signals, such as the Request (REQ#) and Grant (GNT#) signals used for arbitration. In many cases these signals are of high interest for analysis, and in order to make them available for the analyzer, these signals can be brought to eight external inputs on pin headers in the front panel.

These external inputs, fully available in the trigger words, can also be used by any user-specific signals.

### Investigation of Software Problems

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 a board or device fails 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 or device caused it and how.

### Automatic PCI Protocol checking

As an optional piggyback module to the PBT-315, a unique piece of hardware called the PBAT - PCI Bus Anomaly Trigger - is offered. This automatically looks for hardware errors in a PCI system by watching every bus cycle during actual operation. It has rule-based trigger elements that continuously and simultaneously screen all PCI signals, looking for a number of timing and state violations of the PCI protocol. When a violation is found, a message is written on the "Violation screen" and a trigger output signal is generated to cross-trigger the analyzer or to trigger an external instrument like an oscilloscope. Also, a detailed explanation that corresponds to rules given in the PCI Specifications is available for each violation. This will assist the user understanding and correcting the problem.

### Statistics of System Performance

The PCI analyzer may also be used to look at the performance of a PCI bus system. For this purpose, the PBT-315 Bus Analyzer system is equipped with a Statistics module that contains a number of real-time counters controlled by the event word recognizers. This 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.

### Event counting

The Event Counting function, which is based on HW counters, provides a real-time count of the occurrence of four user-defined events. This very powerful function may for example be used to count the number of e.g. IACK cycles per second displayed as a function of time, or to investigate access patterns to the bus in multi-processor systems, etc.

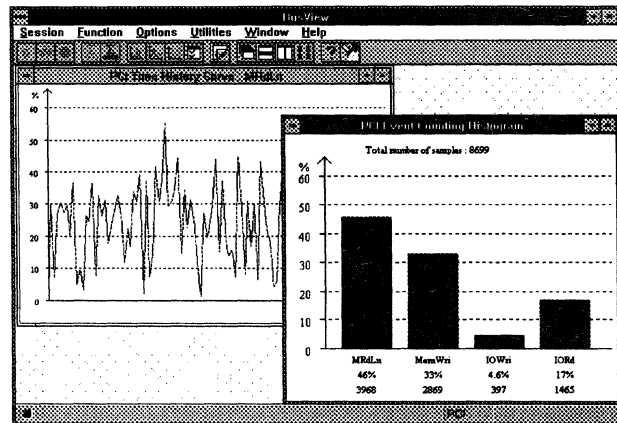


Figure 6 : The STATISTICS functions gives valuable information about PCI performance and traffic patterns.

## **Bus Utilization**

The Bus Utilization function provides a direct readout of the percentage of time the bus is occupied. This is ideal to determine whether the system bus has spare capacity to support another I/O-device or processor etc.

This function, which is based on hardware counters and a pre-programmed usage of the word recognizers, provides an immediate response readout of how the bus is being used at any time.

## **Bus Transfer Rate**

The Bus Transfer Rate function presents how much data is transferred over the bus, shown as MBytes/s and in Mtransfers/s. This can either be shown between selected lines directly in the trace buffer, to measure burst transfer rate, or as histograms that show the average transfer rate over a certain period of time.

This function can be used to characterize a system, to verify if performance specifications have been fulfilled and to assist in system tuning.

## ***Conclusion***

In this paper we have discussed how PCI and PMC Bus Analyzers can be a very powerful tool in detecting, locating and fixing different kinds of PCI bus system problems. These cover all aspects of systems debugging, from low-level hardware to complex software problems. In addition, remarkable statistics functions offers performance measurement functions that allow for system tuning.

Altogether, the PBT-315 PCI Bus Analyzer and the PBTM-315 PMC Analyzer constitute very important tools when building and integrating PCI bus based systems. In many cases these tools could save a very considerable amount of time in debugging such systems, as well as greatly simplify the quality of testing.

PCI: THE BUS THAT GLUES?  
Mark Bronson  
Aeon Systems, Inc.  
8401 Washington Pl. N.E.  
Albuquerque, NM 87113 USA  
(505)828-9120/9115 (fax)  
e-mail: bronson@aquilagroup.com

## **ABSTRACT**

PCI has exploded with a diversity of boards, chips, and systems that promote high performance inter-operability. While there is, indeed, much improved inter-connection with respect to earlier busses, not all performance expectations will always be met. PCI offers a number of "tuning" opportunities that are available on a per device basis. This paper presents design experiences in developing a complex system where PCI bridges and devices are used. An order of magnitude change in system performance by changing configuration fields is analyzed, and some general implications in the use of PCI are discussed.

## **INTRODUCTION**

PCI is unprecedented as a bus standard in the rapidity and diversity with which both board and component implementations are being developed and offered for general consumption. This encourages both vendors of add in boards and developers of stand alone product that need to integrate multiple devices. While delivering an ability to glue, PCI does not always deliver the anticipated performance. A naive expectation of observing the touted bandwidth (132 Mbytes/second) is frequently dashed. Presented here are some experiences in creating a processor module that includes two processors (ALPHA AXP and i960), two bus bridges (to secondary PCI and VMEbus), and I/O devices (Enet, SCSI). There are three major considerations: 1) the basic implementation and "glue" necessary to integrate the components, 2) the interconnect facilities offered by the bridges and how they aided/imposed our system design, and 3) system performance, in particular how PCI "tuning" parameters can result in a data rate change from 4 Mbyte/second to 60 Mbyte/second.

## **HARDWARE IMPLEMENTATION**

One point of consideration for developing a PCI-bridged system is implementation of the hardware. Figure 1 shows a block diagram of the module. The components attached to the PCI include: 1) ALPHA Host memory bridge, 2) PLX to i960 bus bridge, 3) PCI to VMEbus bridge, 4) PCI to PCI bridge, 5) SCSI, and 6) Ethernet.

**ALPHA Host memory bridge.** This provides bi-directional access to the ALPHA processor and memory subsystem. Read/writes mastered external to the ALPHA processor are bridged into the 128 bit memory bus. The bridge maintains cache coherence between the ALPHA secondary and primary caches and the contents of memory.

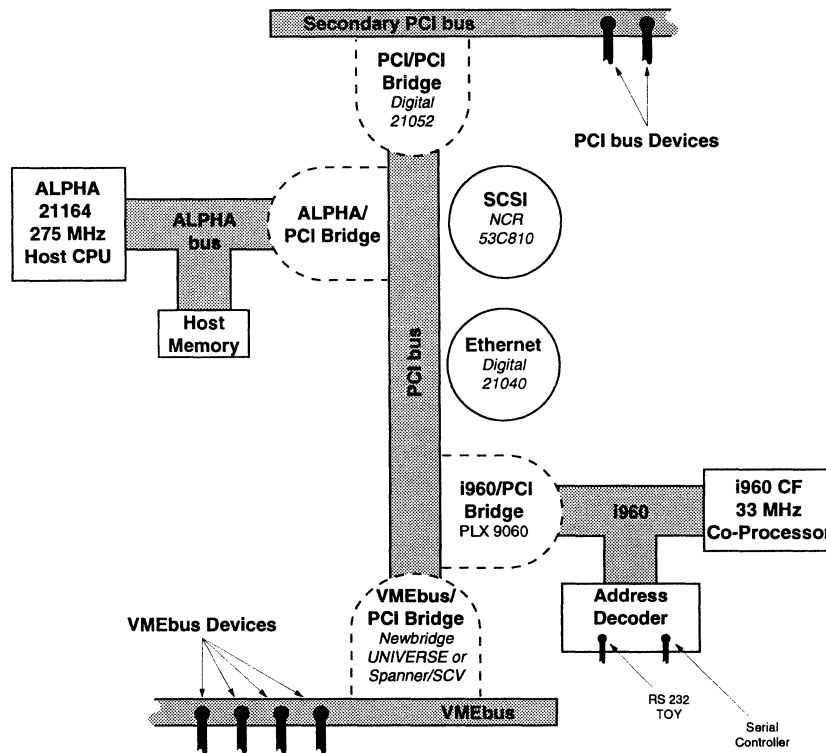
**PLX to i960 bus bridge.** During system design, a performance mismatch was recognized between the ALPHA (which executes at 275Mhz) and the low level I/O frequently required with VMEbus devices. Thus the i960 was included as an I/O Co-processor. To accomplish this task requires full access to the PCI bus, as low level system devices are on the i960 bus (e.g. RS232 console) the bridge must also allow slave access from external masters (e.g. ALPHA CPU).

**PCI to VMEbus bridge.** There have been two implementations used. Initially the combination of PCI to 68040 bridge (Newbridge Spanner) connected to the Newbridge SVC VMEbus controller. A second pass design used the Newbridge UNIVERSE, a direct PCI/VMEbus bridge.

**PCI to PCI bridge.** In order to provide the option to support additional PCI devices, a bridge can be added. To date this has been used to add both custom devices and, in conjunction with a second bridge to connect devices in a PMC form factor.

**SCSI.** NCR PCI/SCSI integrated controller.

**Ethernet.** Digital 21040 integrated Ethernet/PCI controller.



System Block Diagram  
Figure 1

## Putting The Pieces Together

At the first level, all components are integrated as would be anticipated from compliance to the PCI specification. The devil is in the details. An implicit assumption is frequently made by PCI designers that the end system will be a PC with the ancillary defined infrastructure. For example, including a "standard" interrupt controller is not the most effective solution for the very high performance ALPHA; requiring extra external cycles to the interrupt controller impedes the ALPHA in performing useful computation. In addition, in this system, some interrupts may be routed to the i960 (and this decision may be dynamic). Our solution is to make a combination of direct connects to the ALPHA interrupt lines and to incorporate an interrupt controller on the i960 bus. A problem related to interrupts is mediating between the requirements of the various protocols. VMEbus defines seven levels of interrupt priority with a per device vector for multiplexing. Solutions to this problem depend upon a combination of 1) evaluating the mechanisms embodied in the bridge and 2) melding them with the facilities available in the design. The major interconnections (i.e. address/data and control signaling) did just glue together and inter-operate. In total, the necessary logic was implemented in a single PCI arbitration PAL and an interrupt controller. The latter is connected to the i960 bus. Integration of i960 memory and slave devices required a single additional PAL implementing address decode and device acknowledge signals.

## BRIDGE FACILITIES

When designing a bridge, consideration must be given to what overall facilities will be provided. Unlike a PCI/PCI connection, bridges to other bus protocols must provide a translation and, to be effective, some control of the translation to allow each user to make the most effective connection. In addition, the generic assumptions suitable for a straight forward PCI/PCI extension are not sufficient. If bridging to a processor bus (e.g. i960), one needs to assume that a CPU could be included, either as primary or auxiliary system host. The VMEbus is even more drastic; it has long been used to support multiple hosts, which need to be gracefully integrated to the PCI. The facilities of ALPHA host bridge, PLX 9060, and Universe provides a basis for comparison:

	Alpha (host memory)	PLX9060 (i960)	21052 (PCI/PCI)	Universe (VMEbus)
<b>Number of Windows</b>	2	1 (plus expansion ROM)	3 (1 I/O 1 coupled memory 1 prefetch)	9 (5 PCI ->VMEbus 4 VMEbus -> PCI)
<b>Address Translation</b>	Direct or Scatter/Gather	Mask	None	Adder
<b>DMA Engine</b>	No	Yes	No	Yes
<b>FIFO Decoupling</b>	Yes	Yes	Yes	Yes

Table 1: Comparison of ALPHA host bridge, PLX 9060, and Universe Facilities

Address translation is required when processor(s) on the target bus have their own address map. Multiple windows aid by offering simultaneous protocol conversion profiles between the PCI and dissimilar target bus.

### Utility of Multiple Windows

The multiple windows provided by the Universe was a major advantage as was the addition-based address translation. There are several benefits to having a larger number of windows. One is that multiple windows allow greater flexibility in overcoming the difficulties inherent in connecting multiple 32 bit address spaces. As most devices currently available only support a 32 bit address space, not all portions of all buses can be constantly visible. Occupation of address regions on the various buses may be quite sparse. On the i960, this is true in order to take advantage of the bus segments within the processor. In our design, local memory (SRAM), EEPROM, and devices each have different control protocols. The i960 can be programmed to exhibit different bus behaviors, based upon 256Mbyte segments. As a result, the i960 address decode covers small portions of a large (3 x 256Mbyte) address region. VMEbus presents similar difficulties for different reasons. Since the goal was to create an addition for existing VMEbus systems, fewer boundary conditions are better. Two large windows (512 Mbyte) from PCI to VMEbus were set up in the top of PCI address space, one with zero offset (translating directly to the top of VMEbus address space) the other with an offset that makes it zero-based within the VMEbus. One window is configured during system initialization (based upon system specific parameters). An additional window is provided for "on the fly" connection to the rest of VMEbus space. Multiple windows also allow access in both PCI I/O and memory space. This can be very useful in extending the 32 bit address range as some portions can be placed in I/O space. This is appropriate when mapping in I/O devices, where the facilities of memory space (e.g. prefetch) could not be utilized.

### Window Profiles

Bridging to the VMEbus poses difficulty in exporting to PCI the various features that are exclusive to VMEbus. These include three address spaces (A16, A24, and A32), several transaction types (Block and Single Cycle), etc. Two mechanisms were provided within the Universe, and both are convenient. The most obvious is allowing a per window profile, such that PCI transactions are forced to translate to the target protocol type. The second causes portions of a window to have specific attributes (e.g. placing the 64Kbyte A16 region at the top of an A24 region).

### Mechanisms for Address Translation

The three bridges offer three solutions from sophisticated scatter gather to direct mask. Only the ALPHA offered , or really requires, scatter gather. Since the ALPHA executes operating systems that use virtual memory, a contiguous application buffer will not map to physically adjacent pages. While convenient for the user, maintaining the scatter/gather entries greatly increases software complexity. In addition, it adds a performance burden, as the translation must be fetched from memory. This is mitigated in the ALPHA bridge with an on-chip lookaside cache at the cost of increased chip complexity. Most requirements can be met with a simple offset translation, but the adder mechanism is much more convenient and flexible than a mask. In the latter, address lines on the target bus are replaced with a specified pattern. This restricts both the decode flexibility (as the bits replaced are also those used in decode) and precludes most "on the fly" dynamic changes. The base, bound, offset triple of the Universe allowed a number of powerful software facilities to be created for the user.

### Interrupt Handling

As noted earlier, interrupts can generally be a problem when using PCI in a specialized configuration. It is made more difficult when the interrupt protocols are very different. To meet the requirement of full VMEbus support, it was necessary to support all 7 VMEbus levels and to create an acknowledge cycle to retrieve the interrupt vector. In the initial implementation with the SPANNER (PCI to 68Kbus), there was a single interrupt signal external to the bridge with a set of registers to create the per level IACK. This precluded easy support of hierarchical interrupts (i.e. allowing a higher priority interrupt to suspend a processing of a lower priority). The Universe enhanced interrupt support with multiple external lines and an internal mapping to

allow flexible distribution. For our purposes, we route each interrupt line independently to an interrupt controller, allowing preemption. A different setup would funnel all interrupts to a single external line.

### Flexibility is Good

The conclusion: when designing a bridge that is targeted to a broad market, avoid preconceptions of how the end systems will be structured. In particular, avoid taking too centric a view; perspective from both sides of the bridge must be maintained.

### TUNING THE SYSTEM

The PCI specification provides a number of mechanisms for adjusting system performance (e.g. Latency timer, burst control, etc.) When first considering working with PCI, an excitement is generated by the repetition of theoretical bandwidth (132 Mbyte/seconds). It is critical to realize how rapidly this can be squandered. To obtain any significant fraction of the bandwidth, it is essential that (reasonably long) burst transfers be supported, and that they be full speed (i.e. transfer on each clock). Table 2 shows bandwidth as a function of burst length:

<b>Burst Size (dwords)</b>	1	8	16	32	64	256
<b>% of Theoretical Bandwidth (Write)</b>	33.3	80	88.9	94.1	97	99.2
<b>Bandwidth (Write) Mbyte/sec for 33Mhz bus</b>	44	105.6	117.3	124	128	130.9

Table 2: Throughput as a Function of Burst Length<sup>1</sup>

Support of coherent, hierarchical memory systems will usually induce a long latency when reading, but may be able to deliver one or more cache lines thereafter. (In the case of the ALPHA host bridge, the latency for first read can exceed 21 PCI clocks. Once started up to 16 Dwords can be transferred in no wait data beats.)

### Example: Write to VMEbus

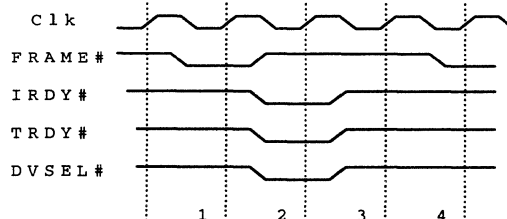
As a concrete example, when first evaluating the DMA performance of the Universe, transferring data from ALPHA host memory to the VMEbus the data rate was ~4 Mbyte/second. Analysis showed that the culprit was a result of the extended delay on starting the pipe into the ALPHA memory system. When a read request is received, the ALPHA bridge probes cache and forces consistency with memory. Then up to two 128 bit transfers (total of 16 dwords) are transferred into a buffer for response to the PCI master. With the Max Latency field loaded with too low a value, the Universe, after an ~20 clock delay, performed two data transfers and terminated the cycle (as required by the specification). The result: two useful cycles in ~24 (counting device select and arbitration overhead). Increasing the timer to a larger value resulted in an increase to 27 Mbytes/second. Modifying the previous table to allow for the delay to first access, we obtain:

<b>Burst Size</b>	1	2	8	16
<b>Bandwidth (Mbyte/sec)</b>	6	11.5	34	56

Table 3: Throughput vs. Burst Length in the Presence of Delay to First Data Transfer

<sup>1</sup>

The single data phase write transaction is presented in the following figure. We assumed that master has already been granted to the bus and that the bus was in Idle state, so there is not any delay due to bus arbitration. Target for this transaction has fast DEVSEL# signal timing and no wait states were involved in the transactions. By not taking into account possibility of back-to-back write transactions on PCI bus, the next transactions on the bus may be started at cycle 4. Therefore, single longword write transaction requires 3 periods of bus clock to be performed and real performance in this case will equal one third of the theoretical maximum. For 33Mhz, the bus maximum is 132Mbyte/sec and real bandwidth is 44Mbyte/sec.





The next measurement of the PCI performance showed that, while the Universe was willing to continue the burst, the ALPHA bridge was terminating it at 8 cycles. Changing the configuration of the ALPHA bridge to permit 16 dword long bursts increased the PCI performance to the expected 56 Mbytes/second (where VMEbus becomes the throttle).

### **Implications of needing Burst Data**

This example demonstrates the enormous impact that tuning the system can have, it also indicates the need for long bursts. Burst transactions will be the only method to maintain performance when using PCI/PCI bridges. The convenience of the bridge, in combination with the structural restrictions of the PCI (i.e. number of possible nodes and line lengths), makes them appear a panacea. There is a substantial delay for transactions to transit the bridge (5 clocks on the Digital 21052). This delay does not affect all transactions equally; posted writes may see very little delay while single cycle reads will suffer the full effect. Again, the only general approach is to ensure that transfers are multi-word bursts. A direct consequence of requiring data to be transferred as larger blocks, is that subsystems need to have internal buffer resources and probably local intelligence to ensure full utilization of the buffer. This is in direct contrast to many previous protocols where the intention was to concentrate the intelligence at a single host. Fortunately, the rapid proliferation of low cost, extremely powerful engines (e.g. i960, ARM, 68K etc.) make distributing intelligence possible. The burden on software remains to be addressed.

### **I/O Co-Processor**

In our system, the i960 acts as an I/O Co-processor, allowing it to consolidate data transfers for the ALPHA host. In this implementation, the serial lines (which reside on the i960 local bus) can be turned from very high overhead single byte devices to a more efficient, line oriented protocol. While very advantageous to the ALPHA, the small grain transactions performed by the i960 still adversely affect PCI bandwidth. To fully realize the potential of the i960, there needs to be a decoupling bridge between the primary PCI and a secondary PCI on which devices reside. Announced products (e.g. Intel i960RP) will deliver exactly this capability in a single package.

### **CONCLUSION**

Designing around PCI greatly reduces the effort required to "glue" components together. Given the relatively recent creation of the specification the number of already available parts and vendors is remarkable. Interest in PCI is increasing very rapidly; there is a continuous litany of new announcements for new parts that attach to PCI, or bridges to directly connect existing processors into a PCI system. Simplifying the implementation process does promote the potential to overlook system performance issues. Without careful analysis of data flow, device PCI usage characteristics and queuing delays a very misleading assumption of final performance will be made. When considering a device's PCI performance it is important to look at both the direct transfer rate (number of cycles between each data transfer within a burst) and the delay incurred in starting a transaction. The latter is most likely to be an issue on reads from a memory system (e.g. across host processor bridge). As processing power is packaged for PCI, distribution of intelligence will become more prevalent. Using this in any kind of a generic (i.e. standard) manner will pose a very interesting challenge. In the meantime specialized configurations will add to the understanding and interest in the potential that PCI brings to system implementation.

### **BIOGRAPHY**

Mark Bronson is the Director of Research and Development for Aeon Systems, Inc. (Aeon). As a wholly-owned subsidiary of Aquila Technologies Group, Inc., Aeon is engaged in the design and manufacture of products for distributed, real-time data acquisition/control and surveillance. Aeon is on the leading edge of real-time instrumentation and connectivity technology. Of VME processor boards based on the DEC Alpha RISC processor, Aeon offers the fastest implementation.

Mark Bronson is responsible for the development of new projects and products from conception to production. His experience includes a wide range of software systems (UNIX, MACH, OSF/1, VMS, and real time kernels) implemented on a variety of processor architectures (VAX, Alpha AXP, MIPS, and 68K among others) and bus standards (VMEbus, MBII, PCI, BI, etc.). Under his guidance, development efforts at Aeon have expanded from an exclusive VAX/VMS focus to an integration of distributed (networked) open systems running on a variety of standard platforms.

# LATENCY ISSUES IN POWERPC REFERENCE PLATFORM ARCHITECTURES

Don Dingee  
Product Marketing Manager  
Motorola Computer Group  
2900 S. Diablo Way DW212  
Tempe, AZ 85282 USA  
(602) 438-3034  
dad@phx.mcd.mot.com  
<http://www.mot.com/computer>

## ABSTRACT

Published results for MPC105 (also known informally as "Eagle") throughput performance are available but cache line latency performance is less clearly deterministic and more problematical. This paper defines important issues, illustrates a range of cases with specific numerical results, and discusses a true worst case and its preclusion.

## INTRODUCTION

An article by Wang et al appearing in the April 1995 issue of IEEE Micro describes the maximum PCI-to-system memory throughput numbers for an MPC105 PCI bridge/memory controller implementation.<sup>1</sup> That article makes many simplifying assumptions about latency issues. For instance, it states "... A PCI master can read from system memory at a data transfer rate of 9-1-1-1" PCI clocks. Actual performance may be affected by time to gain mastership of the system memory bus, time associated with snooping memory to maintain cache coherency, and time to transfer the first beat of data. Further investigation is needed to bound and characterize performance.

This paper discusses the specific PowerPC 603/603e/604 microprocessor and MPC105 PCI bridge/memory controller implementation found in Motorola PowerPC reference platform products such as the MVME1603, MVME1604, MVME1300, Ultra, Atlas, and Chameleon. It backgrounds the basic issues of latency and throughput. It clarifies the important parameters, and describes the best, typical, and worse or worst cases involved in computing figures of merit. During this discussion, a worst case scenario is described, and a range of potential remedies to preclude worst case are proposed. Finally, the results are summarized and some general conclusions are drawn.

## LATENCY AND THROUGHPUT BACKGROUND

The PCI bus master to system memory transfer latency and throughput are points of primary concern in real-time embedded system design. Table 1 summarizes the related MPC105 PCI performance parameters.

Table 1: MPC105 PCI Bus Performance

<i>Transfer type</i>	<i>PCI clocks (33 MHz bus)</i>
Single read (4 bytes)	9
Read line (32 bytes)	9-1-1-1-1-1-1
Read multiple (successive 32 byte bursts)	9-1-1-1-1-1-1/4-1-1-1-1-1-1/4-1-1-1-1-1-1/...
Single write (4 bytes)	2
Write line (32 bytes)	2-1-1-1-1-1-1
Write multiple (successive 32 byte bursts)	2-1-1-1-1-1-1/2-1-1-1-1-1-1/2-1-1-1-1-1-1/...

<sup>1</sup> Karl Wang et al, "Designing the MPC105 PCI Bridge/Memory Controller," IEEE Micro, April 1995, pp. 44-49.

Another important MPC105 parameter is the PowerPC local bus performance. On a burst transfer to local DRAM with 60 nsec access time, with a local bus speed of 66 MHz, the PowerPC 603/603e/604 transfers four 64-bit double words (total of 32 bytes) in 20 clocks (8-4-4-4) for a total transfer time of 0.300  $\mu$ sec.

From this data, the limiting factor in PCI to system memory performance is the PCI read, which is studied in more detail in the following discussion.

In a high performance system, PCI reads are 32 byte burst transfers (8 beats of 4 bytes each) by PCI bus mastering devices. Latency and throughput for a given 32 byte burst can be defined by examining the states of the transfer and stating assumptions about those states. A PCI read of DRAM proceeds as follows:

*PCI bus master presents addresses to MPC105*

- Assume no other PCI traffic, and PCI bus master is highest priority PCI requester and is not held off.

*MPC105 removes local bus grant to PowerPC processor*

- Time to remove grant is insignificant.

*MPC105 gains local bus mastership*

- PowerPC processor can complete up to 2 four beat (32 byte) external transactions to local DRAM after its grant is removed.

*MPC105 snoops transaction to ensure cache coherency*

- Snoop collision may occur, caused by internal resource conflict in PowerPC processor and signaled by ARTRY. Collision is retried every 4 local bus clocks until resolved. Most collisions resolve after one retry.

*MPC105 completes burst read of 32 bytes from DRAM and forwards transfer onto PCI master*

- MPC105 has 32 byte read buffer, which does not have to be filled prior to data being presented to PCI bus. Once burst begins, it proceeds to completion.
- Assume no other PCI traffic, and PCI transaction finishes concurrently with DRAM read with no additional overhead.

Given this transfer sequence, latency and throughput are defined as follows:

$$\text{PCI read latency} = t_{\text{PRL}} = t_{\text{LBM}} + t_{\text{SRC}} + t_{\text{T32B}}$$

where:

- $t_{\text{LBM}}$  = time for MPC105 to gain local bus mastership,
- $t_{\text{SRC}}$  = time for MPC105 to snoop and resolve collision,
- $t_{\text{T32B}}$  = time for MPC105 to transfer 32 bytes to PCI.

$$\text{PCI read throughput for 32 bytes} = 32 / t_{\text{PRL}}$$

With these definitions, each parameter can be examined in more detail in best, typical, and worse or worst cases, and figures of merit for latency and throughput derived.

### **LOCAL BUS MASTERSHIP**

In the best case, the MPC105 already has mastership of the PowerPC local bus, and no other contention occurs, so  $t_{\text{LBM}} = 0$ . This would be the case in the steady state portion (after the first cycle) of a PCI DMA transfer.

In a typical case, assuming a 70% processor hit rate in cache, the MPC105 has a 30% chance of waiting for the processor to complete a single external burst transaction. Previously, the time for a processor burst transfer to local DRAM was shown as 0.300  $\mu$ sec. This creates a typical  $t_{\text{LBM}}$  over a relatively large data transfer of 30% of 0.300  $\mu$ sec, or 0.090  $\mu$ sec.

In a worst case, the MPC105 is held off while the processor completes two burst transactions, so  $t_{LBM} = 0.600 \mu\text{sec}$ . This is an upper bound on  $t_{LBM}$  given the 60 nsec DRAM access time assumed earlier.

### ***SNOOPING AND RESOLVING COLLISIONS***

The best case for a snoop is no collision, or  $t_{SRC} = 0$ .

The PowerPC 604 microprocessor implements dual-ported cache address tags on its L1 cache, and allows snooping to occur concurrently with cache accesses. This improves performance and minimizes collisions to situations where cache buffer cast-outs are required to maintain coherency. The current PowerPC 604 microprocessor has only one cast-out buffer, so only one cast-out can be pending.

The PowerPC 603/603e microprocessor implements single-ported cache address tags on its L1 cache, and thus presents several snoop collision scenarios. These include snoop hits during a burst load operation, snoop hits while a cast-out is pending (one cast-out buffer is present), and snoop attempts while the L1 cache is being accessed by a load or store operation.

The MPC105 retries a collided snoop every 4 local bus clock cycles, and most collisions resolve after one retry. Collisions occur rather infrequently, so typically  $t_{SRC} \cong 0$  over a relatively long transfer.

A worse case for a normal snoop collision is when a cast-out is required, and the cast-out data phase overlaps the previous burst. ARTRY is signaled after the first beat (8 clocks) of the cast-out burst. A worse case for a snoop hit followed by a burst write is very similar in overall timing. The PowerPC split transaction bus allows transaction overlap, but in general, the worse case snoop latency under normal circumstances is 36 clocks (remaining 3 beats of transaction at 4 clocks each, plus another transaction at 8-4-4-4, plus one more MPC105 retry). This puts a worse case  $t_{SRC} = 0.540 \mu\text{sec}$ . This is normally also the worst case.

A true worst case, with remedies to preclude it, is discussed later.

### ***TRANSFERRING PCI DATA***

Once the MPC105 actually begins to transfer data, it completes and there is no other PCI overhead involved.

Best case for the PCI read of 32 bytes is when the speculative read feature of the MPC105 is enabled, and a multiple read with successive bursts is in progress. As described previously, this results in a transfer time for 32 bytes of 11 clocks (4-1-1-1-1-1-1) so  $t_{T32B} = 0.333 \mu\text{sec}$  on the 33 MHz PCI bus.

Typical and worst cases for the data transfer time are 16 clocks (9-1-1-1-1-1-1) so  $t_{T32B} = 0.485 \mu\text{sec}$ .

### ***FIGURES OF MERIT***

Given this discussion of local bus mastership, snooping and collision resolution, and PCI read transfer time, the latency and throughput formulas can be used to develop the figures of merit shown in Table 2.

**Table 2: PCI Read Figures of Merit**

<i>Case</i>	$t_{LBM}$ ( $\mu\text{sec}$ )	$t_{SRC}$ ( $\mu\text{sec}$ )	$t_{T32B}$ ( $\mu\text{sec}$ )	<i>Latency</i> ( $\mu\text{sec}$ )	<i>Throughput</i> ( <i>MB/sec</i> )
Best	0	0	0.333	0.333	96.0
Typical	0.090	0	0.485	0.575	55.7
Worse / Worst	0.600	0.540	0.485	1.625	19.7
Worse/worst, with low priority PCI transfer in progress before start of PCI read				3.250	9.85

Although the PCI read is a relatively low priority transaction on the processor local bus (page 8-8 of the MPC105 User's Manual lists it as priority 11), most of the other higher priority local transactions have been accounted for in the worse cases in the form of up to two PowerPC microprocessor external burst transactions in our assumption. L2 cache would affect this model somewhat, but has not been considered in this analysis.

If a lower priority PCI master were to have a transfer in progress, a high priority PCI read discussed here would be held off by one PCI burst transaction. The worse/worst case discussed would double in latency and halve in throughput, as shown in the last line of Table 2.

### ***TRUE WORST CASE SNOOP COLLISIONS***

A pathologically worst case problem has been identified as a remote possibility on the PowerPC 603/603e microprocessor due to its single-ported L1 cache address tags. On the PowerPC 603/603e, a collision is signaled if a snoop is attempted while the L1 cache is being accessed by a load or store operation. The MPC105 retries a collided snoop every 4 clocks until successful. If the MPC105 snoop retry period of 4 clocks were aligned with a cached loop writing to the L1 cache address tags on the same 4 clock timing, snoop collisions would potentially occur until the cached loop terminated.

Writing the cache address tags, not just reading them, creates the potential for this scenario. Cache control instructions which write the tags are the user instruction DCBZ (data cache block zero) and the supervisor instruction DCBI (data cache block invalidate).

An example would be executing a cached loop consisting of a DCBI followed by an ADD to a counter and a BCND. This instruction sequence would write the cache address tags 1 out of every 4 clocks. An instruction sequence such as this is not particularly useful and is not considered likely to occur, but nonetheless is a possibility and must be considered.

Table 3 shows the effect on typical latency and throughput for this situation. The snoop collisions are assumed to occur on every retry by the MPC105, adding 4 local bus clocks (60 nsec) to the latency for every collision.

**Table 3: Repeated Snoop Retries, Typical PCI Read Case**

<i># Snoop Retries</i>	<i>Latency</i> ( $\mu\text{sec}$ )	<i>Throughput</i> ( <i>MB/sec</i> )
0	0.575	55.7
4	0.815	39.3
16	1.535	20.8
64	4.415	7.3
128	8.255	3.9

As shown by Table 3, should a long loop of this true worst case occur, latency can be severely impacted. What begins as good performance in the typical case degrades severely under these circumstances.

There are many possible remedies to this behavior. Disabling the MPC105 snoop capability (by not asserting the GBL signal) would prevent snoop collisions altogether, but would severely impact performance by introducing software overhead to maintain cache coherency. Implementing the PCI lock protocol and executing longer PCI transfers would reduce the impact of a single collision episode and increase average throughput. However, these hardware approaches are not generic and have other undesirable performance impacts.

Truly precluding this worst case behavior in a general way requires a software approach, and there are two solutions which are plausible.

First, all data could be marked global, or if not global at least cache-inhibited and guarded (see the WIMG bit definitions of the PowerPC 603 and PowerPC 604 User's Manuals). This forces the processor to execute external cycles. During a snoop collision the MPC105 is local bus master, and the processor must wait to execute the external cycle. Most snoop collisions are prevented, and any snoop collision that does occur is resolved on the first retry. This impacts CPU performance by reducing the data cache hits, and is a severe workaround.

The best solution is to ensure that use of the cache control address tag write instructions, DCBZ and DCBI, do not occur in loops of 4 or 8 clocks. This can be done by avoidance of these instructions, or by insertion of no-op instructions to change the loop timing. This results in snoop collision resolution on the first retry.

#### ***SUMMARY***

Motorola PowerPC microprocessor platforms using the MPC105 PCI Bridge/Memory controller can achieve significant real-time throughput with low latency. A limiting case of a PCI read was studied and found to have a typical latency of 0.575  $\mu$ sec and a typical throughput of 55.7 MB/sec. Both best and worse case scenarios were considered to place upper and lower bounds on performance. A truly worst case scenario, while a very remote possibility, was considered for the PowerPC 603/603e microprocessor. A precise workaround involving care with DCBZ and DCBI instructions to preclude the true worst case from occurring was illustrated.

#### ***REFERENCES***

**Karl Wang et al, "Designing the MPC105 PCI Bridge/Memory Controller," IEEE Micro, April 1995, pp. 44-49.**

**Motorola, MPC105 PCI Bridge/Memory Controller User's Manual, MPC105UM/AD**

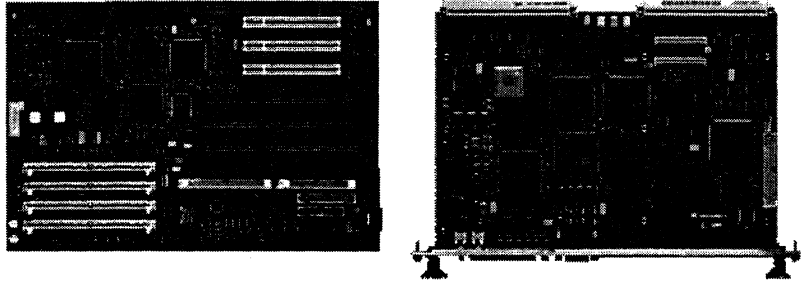
**Motorola, PowerPC 603 RISC Microprocessor User's Manual, MPC603UM/AD**

**Motorola, PowerPC 604 RISC Microprocessor User's Manual, MPC604UM/AD**

***PowerPC is a trademark of International Business Machines Corp.***

**Latency Issues In PowerPC  
Reference Platform Architectures**

**PowerPC™**



**Don Dingee**  
Product Manager  
Motorola Computer Group  
Embedded Technologies Marketing

\*PowerPC and the PowerPC logo are trademarks of International Business Machines Corp.




**Latency Issues in PowerPC  
Reference Platform Architectures**

**PCI Spring '96  
Motherboards and Board Computers**


**PCI Latency and Throughput**

*Motorola PowerPC Board Products*  
<http://www.mot.com/computer>

**VME boards:**  
MVME1600  
MVME1300



**Motherboards:**  
Ultra  
Atlas  
Chameleon



- ❑ Embedded designs usually have real-time response criteria
  - ✓ Latency
  - ✓ Throughput
- ❑ What can PowerPC microprocessors and PCI bus do?
  - ✓ What affects performance?
  - ✓ What is best, worst case?
- ❑ Given microprocessor and chipset parameters, analysis can be done and performance characterized
  - ✓ Best, typical, worse cases
  - ✓ True worst case scenario, remedies
  - ✓ Conclusions



**Latency Issues in PowerPC  
Reference Platform Architectures**

**PCI Spring '96  
Motherboards and Board Computers**

## MPC105 "Eagle" Performance

### **Basic Performance Parameters (33 MHz PCI bus)**

- ▲ Single read (4 bytes):           9 PCI clocks
- ▲ Read line (32 bytes):           9-1-1-1-1-1-1-1-1
- ▲ Read multiple:                   9-1-1-1-1-1-1-1-1/4-1-1-1-1-1-1/...
  
- ▲ Single write (up to 32-bit):    2 PCI clocks
- ▲ Write multiple:                  2-1-1-1-1-1-1-1-1

### **DRAM Bandwidth (66 MHz PPCbus, 64-bit, 60 nsec DRAM)**

- ▲ Burst access:                    8-4-4-4

Limiting factor is PCI read -- case is studied in detail



**MOTOROLA**  
Computer Group

*Latency Issues in PowerPC  
Reference Platform Architectures*

*PCI Spring '96  
Motherboards and Board Computers*

## PCI Read Sequence

- *PCI bus master presents addresses to MPC105*
- *MPC105 removes local bus grant to PowerPC processor*
- *MPC105 gains local bus mastership –  $t_{LBM}$*
- *MPC105 snoops transaction to ensure cache coherency –  $t_{SRC}$*
- *MPC105 completes 32 byte burst read, forwards to PCI –  $t_{T32B}$*

**PCI Read Latency**      $(t_{PRL}) = t_{LBM} + t_{SRC} + t_{T32B}$   
**PCI Read Throughput** =  $32 \div t_{PRL}$



**MOTOROLA**  
Computer Group

*Latency Issues in PowerPC  
Reference Platform Architectures*

*PCI Spring '96  
Motherboards and Board Computers*



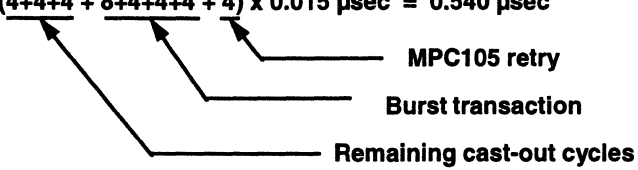
## Local Bus Mastership

- **Best case -- MPC105 already has local bus mastership**
  - ✓  $t_{LBM} = 0$
  - ✓ true in case of ongoing DMA transfer
- **Typical case -- 30% chance of 1 processor cycle**
  - ✓ assumes 70% L1 cache hit rate
  - ✓  $t_{LBM} = 30\% \text{ of } 0.300\mu\text{sec} = 0.090 \mu\text{sec}$
- **Worst case -- processor completes two burst cycles**
  - ✓  $t_{LBM} = 2 \times 0.300\mu\text{sec} = 0.600 \mu\text{sec}$



## Snooping and Resolving Cache Collisions

- **Best case -- no collision**
  - ✓  $t_{SRC} = 0$
- **Typical case -- very low percentages of collisions**
  - ✓ most collisions resolve after one retry
  - ✓  $t_{SRC} = 0$  over long term
- **Worse case -- cast-out overlap with another burst**
  - ✓ collision signaled after first beat of burst
  - ✓  $t_{SRC} = (4+4+4 + 8+4+4+4 + 4) \times 0.015 \mu\text{sec} = 0.540 \mu\text{sec}$



## Transferring PCI Data

Once PCI transfer begins, it completes -- no other PCI overhead

- **Best case -- MPC105 in progress with speculative read**  
 ✓  $t_{T32B} = (4-1-1-1-1-1-1) \times 0.030 \mu\text{sec} = 0.333 \mu\text{sec}$
- **Typical and worst case**  
 ✓  $t_{T32B} = (9-1-1-1-1-1-1) \times 0.030 \mu\text{sec} = 0.485 \mu\text{sec}$



**MOTOROLA**  
Computer Group

*Latency Issues in PowerPC*  
Reference Platform Architectures

PCI Spring '96  
Motherboards and Board Computers

## Figures of Merit

Case	$t_{LBM}$ ( $\mu\text{sec}$ )	$t_{SRC}$ ( $\mu\text{sec}$ )	$t_{T32B}$ ( $\mu\text{sec}$ )	Latency ( $\mu\text{sec}$ )	Throughput (MB/sec)
Best	0	0	0.333	0.333	96.0
Typical	0.090	0	0.485		
Worse/worst	0.600	0.540	0.485	1.625	19.7
Worse/worst, with low priority PCI transfer in progress before start of PCI read				3.250	9.85

- Most other MPC105 transactions accounted for in worst case
- Model does not account for possible effect of L2 cache

**Actual customer data:**  
PMC, DMA read from DRAM  
  
60 MB/sec sustained  
70 MB/sec peak



**MOTOROLA**  
Computer Group

*Latency Issues in PowerPC*  
Reference Platform Architectures

PCI Spring '96  
Motherboards and Board Computers

## True Worst Case and Remedies

- ❑ PowerPC 603/603e
  - ✓ Single ported L1 cache address tags
  - ✓ Collisions if snoop attempted while L1 cache being accessed by load/store
- ❑ PowerPC 604
  - ✓ Dual ported L1 cache address tags
  - ✓ Does not collide on snoop attempt while L1 cache being accessed
- ❑ A very remote possibility on 603/603e system

### Best remedy

- ▲ DCBZ, DCBI instructions write cache address tags and create collision scenario
- ▲ Manage these instructions so they don't occur ...
- ▲ ... or alter timing so they don't beat in 4 cycle loop

### Other remedy

- ▼ Mark data cache-inhibited and guarded to force external cycles
- ▼ Impacts performance, but may be acceptable



Latency Issues in PowerPC  
Reference Platform Architectures

PCI Spring '96  
Motherboards and Board Computers

## Conclusions

- ❑ PowerPC microprocessor and MPC105 can achieve significant throughput
- ❑ PCI read limiting case
- ❑ PCI read latency
  - ✓ Best 0.333  $\mu$ sec, Typical 0.575  $\mu$ sec
- ❑ PCI read throughput
  - ✓ Best 96.0 MB/sec, Typical 55.7 MB/sec
- ❑ True worst case scenario on PowerPC 603/603e is explicitly avoidable
  - ✓ DCBZ, DCBI instructions can be managed
- ❑ Framework developed here can be used to analyze other PCI chipsets and perform comparisons

Actual data:  
60 MB/sec sustained  
70 MB/sec peak



Latency Issues in PowerPC  
Reference Platform Architectures

PCI Spring '96  
Motherboards and Board Computers

# PCI Passive Backplane Technologies

Joe Pavlat

V.P., Product Development  
Pro-Log Corporation

# The Industrial World Needs a Platform:

- Fast Enough to Support new CPU's & Peripherals (>100 Mbytes/sec)
- Supports 15-20+ Plug-in Cards
- Able to Support CPU's from all Major Manufacturers (Intel, DEC, Motorola)

....PCI Meets That Need.

# PCI- Born in the Desktop World

- Developed by Intel
- Embraced by Major Microprocessor Manufacturers:
  - INTEL (Pentium, PentiumPRO)
  - MOTOROLA, IBM (PowerPC)
  - DEC (Alpha)

# PCI - Born in the Desktop World

- Used in over 80% of Desktop PC's (continually increasing)
- Desktop Market >\$150 billion
- PCI Has Won the "Local Bus War"

# PCI - Very High Performance

- 133 MBytes/sec. Transfer Rate (32 bits, 33 MHz Bus Clock)
- Up to 532 MBytes/sec. Transfer Rate (64 bits, 66 MHz Bus Clock)
- Other Buses Much Slower:
  - VME: ~ 40 MBytes/sec.
  - VME64: ~ 80 Mbytes/sec.
  - ISA: ~ 2 MBytes/sec.



# PCI - Very High Performance

- These High Speeds Needed For:
  - Real-Time Video Processing
  - Next Generation, High Speed Networking

*Applications seem to grow to use all available bandwidth.....*

# Desktop PCI Limitations

- Small Number of Expansion Slots (~4)
- Less-Than-Optimal Form Factor:
  - Horizontal, Long, Skinny Cards
  - Small Connector Edge
  - Poor Cooling, Card Retention
  - Hard to Mount in a Cabinet
- Active Motherboards Have High MTTR

**PCI is Great, but.....**

**Desktop Form Factor Not Suitable  
for Many Industrial &  
Telecommunications  
Applications**

# Industrial/Telecom Needs:

- A Bus Fast Enough to Support new CPU's and Peripherals (>100 MB/sec.)
- A Bus Able to Support CPU's from all Major Manufacturers (Intel, DEC, Motorola, Sun, etc.)

# Industrial/Telecom Needs:

- Rugged Packaging:
  - Good Shock, Vibration, Temperature specs
  - Better Cooling
  - Bigger, More Reliable Power Supplies
- Modularity:
  - 15-20 Slots a Typical Requirement
  - Must be Simple to Add Functions

# Industrial/Telcom Needs:

- Better Serviceability - Low MTTR
- More Mounting Flexibility:
  - 19" Rack
  - NEMA Cabinets
- Configuration Control - Can't have the BIOS or Video Chip changing every week.

# Industrial/Telecom Needs:

- **Better Connecting:**

- Industrial User Wants I/O out front - often needs screw terminals
- Telecom User Wants I/O out rear - clean front panels and quick-to-replace modules

Everybody Hates Flat Cables.....

# Industrial/Telecom PCI

- Two Predominant Standards:
  - Passive Backplane (“Slot Cards”)
  - CompactPCI



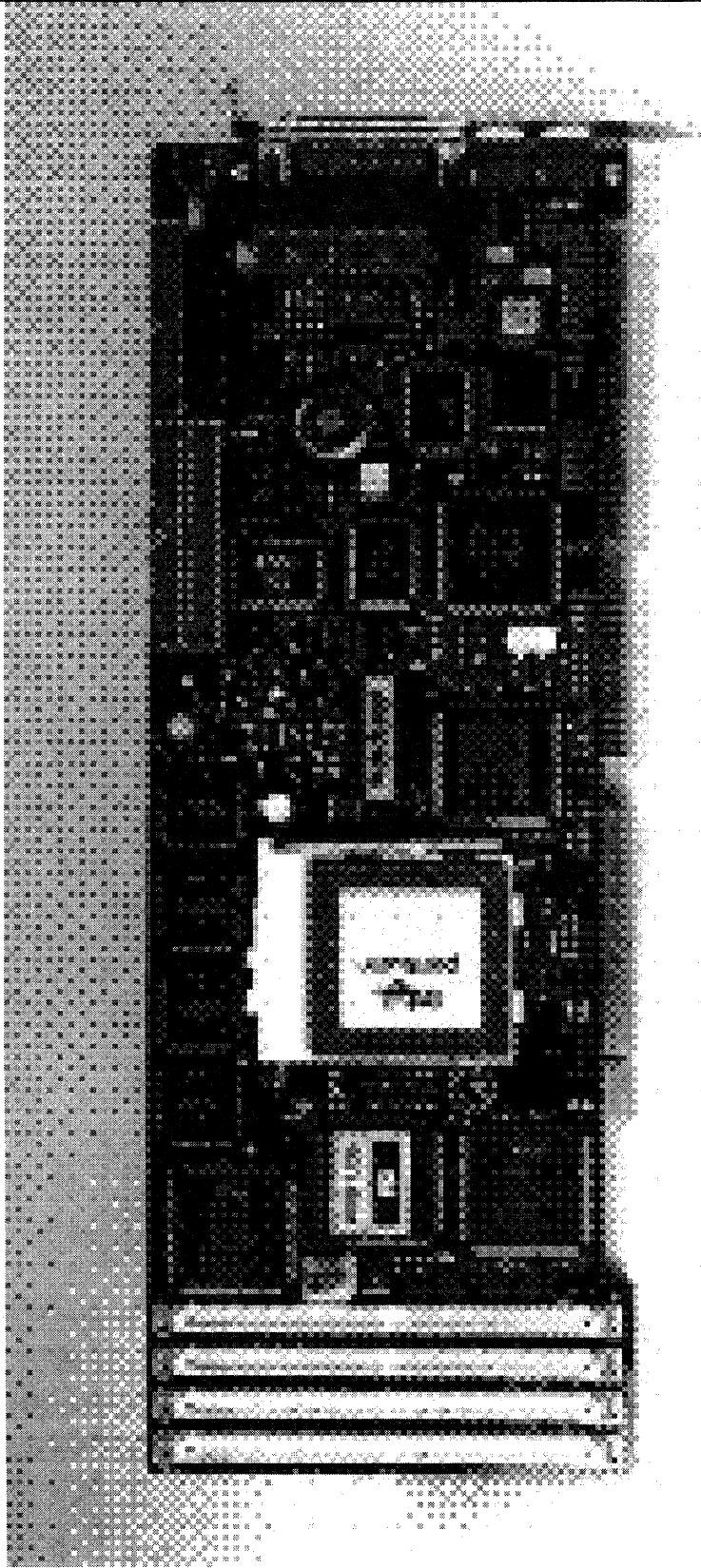
# Passive Backplane PCI

- All Active Circuitry Moved To a Plug-in Card - Has PCI and ISA Buses
- Motherboard Replaced With a Passive Backplane - Connectors Only
- PCI Industrial Computer Manufacturers Group (PICMG) Standard Published 1994

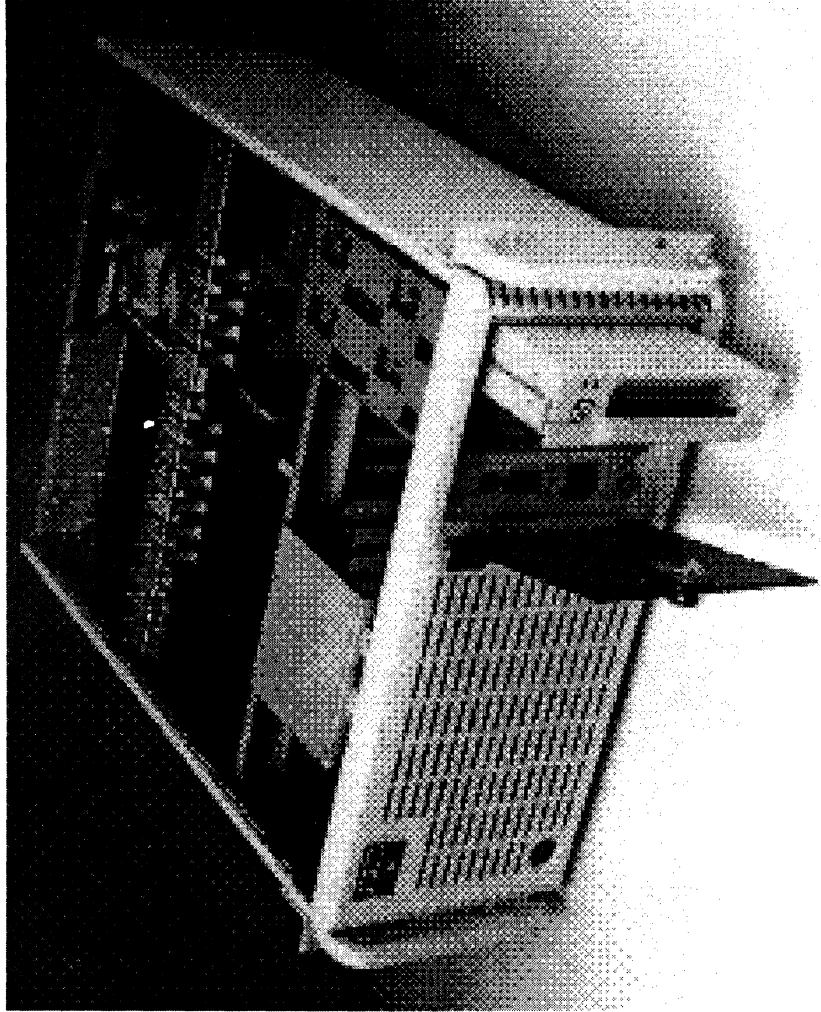
# Passive Backplane PCI

- Usually 4 PCI and 8-15 ISA Slots
- Changing or Upgrading a CPU takes only Minutes
- Rugged Chassis Available:
  - Bigger, Better Power supplies - often Dual
  - Card Retention Mechanism
  - Usually 19" Rack Mount

# Passive Backplane PCI CPU



# Passive Backplane Chassis



# Passive Backplane PCI STRENGTHS

- Low MTTR (~ Minutes)
- Easy to Upgrade CPU
- Can Use Existing ISA I/O Cards
- Rackmount, Desktop, or Tower Packaging

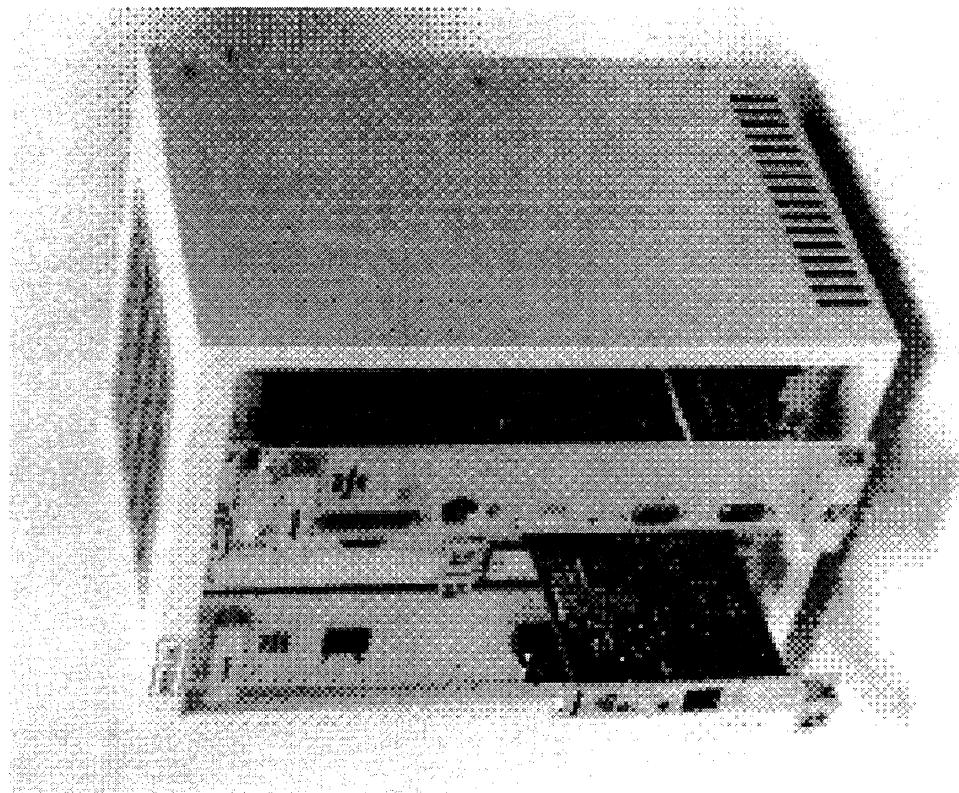
# Passive Backplane PCI WEAKNESSES

- Still Limited to 4 PCI Slots (although this can be expanded with Bridge chips)
- Poor Cooling, Card Retention
- Limited External Connector Area
- Hard to Mount in NEMA-type Enclosure

# CompactPCI

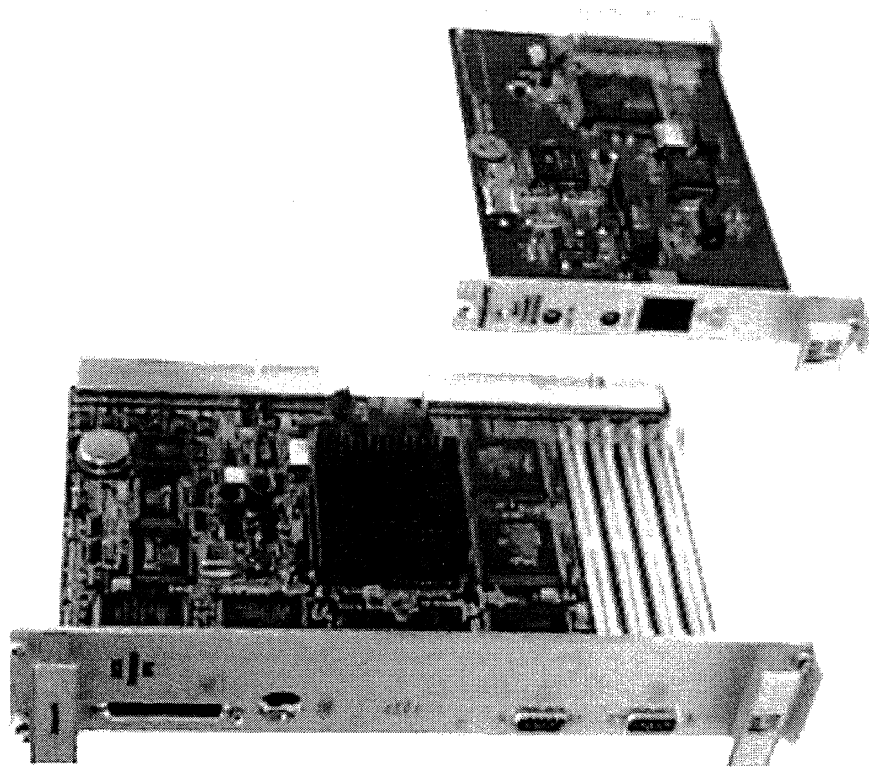
- New Standard (PICMG Approved 12/95)
- Simple Concept: Commercial PCI Silicon in Eurocard Packaging
- Uses High Density Pin-and-Socket Bus Connectors
- 3U (100mm x 160mm) and 6U (233 mm x 160 mm) Sizes Defined

# CompactPCI





# CompactPCI



# CompactPCI - STRENGTHS

- Rugged Packaging
- Packaging Well Accepted - VME Uses
- Excellent Card Retention, Cooling
- Good Shock & Vibration Characteristics
- Can be Easily Rack or Panel Mounted
- I/O out Front or Rear
- 8 Slots (More with Bridge Chips)

# CompactPCI - STRENGTHS

- Connector IEC Standard (IEC-917 & IEC-1076)
- Connector Bellcore Qualified (Tr-NWT-001217)
- Connector Widely Accepted in Europe
- CE Certification Straightforward
- Can Bridge to VME, ISA, STD Buses (Hybrid Systems)

# CompactPCI - WEAKNESSES

- Brand New - Limited Number OF Suppliers Today
- 8 Slot Maximum in Basic Configuration - Needs Bridge Chips to Expand (One Chip Every 8 Additional Slots)

# Driven by Commercial Silicon & Standards

- Commercial Standards have >100 times the Sales of Industrial Standards
- People are Familiar with and Educated About Commercial Standards
- Hottest New Hardware & Software Developed for Desktop PCI First
- Best Software Development Tools

# “I Want PCI Performance - Which Version Do I Choose?”

- Desktop PCI When:

- Initial Hardware Cost is Very Important (i.e., You're Cheap)
- Your Application can Tolerate Lack of Revision Control (unexpected BIOS changes, etc.)
- Desktop Environment (Clean, Cool, etc.)

# “I Want PCI Performance - Which Version Do I Choose)

- Passive Backplane PCI When:
  - Low MTTR Important
  - Simple CPU Changes or Upgrades Important
  - “Light Industrial” Environment

# “I Want PCI Performance - Which Version do I Choose?”

- CompactPCI When:

- Robust Packaging Required
- Front or Rear Panel I/O Required
- Environment Requires Good Shock, Vibration, and Temperature Specs
- Bridge to Bus Other Than ISA Needed:
  - VME
  - STD
  - etc.



# Industrial/Telecom PCI - Conclusions

- PCI Provides Highest Performance, Speed Available Today
- Uses High Volume, Low Cost Silicon
- Leverages Commercial PCI Innovations
- Well Positioned for next 5 Years
- Choice of Form Factors (CompactPCI, Passive Backplane)

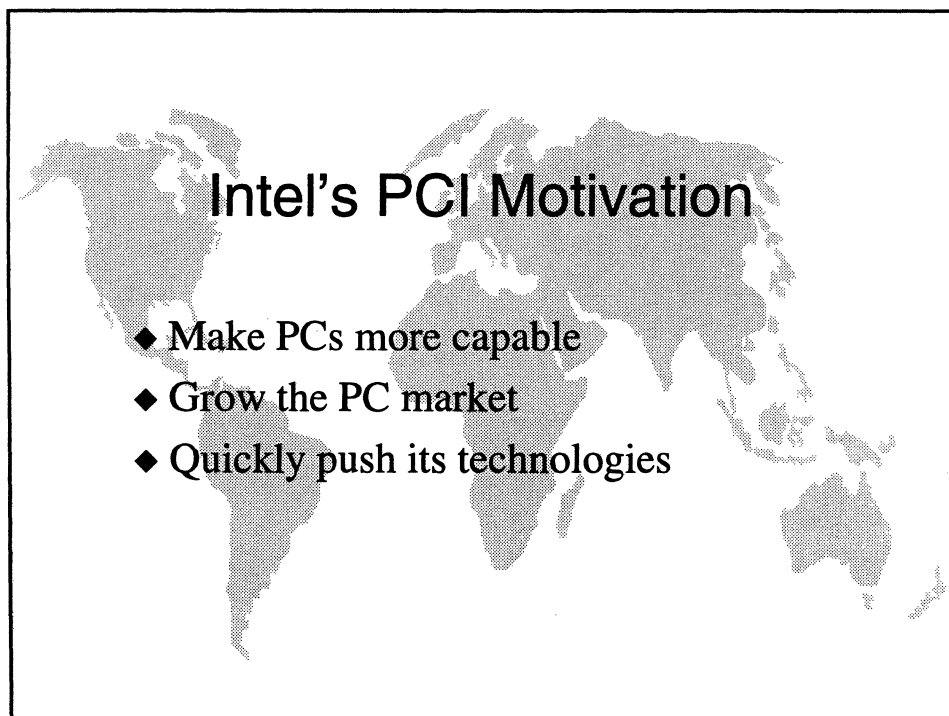
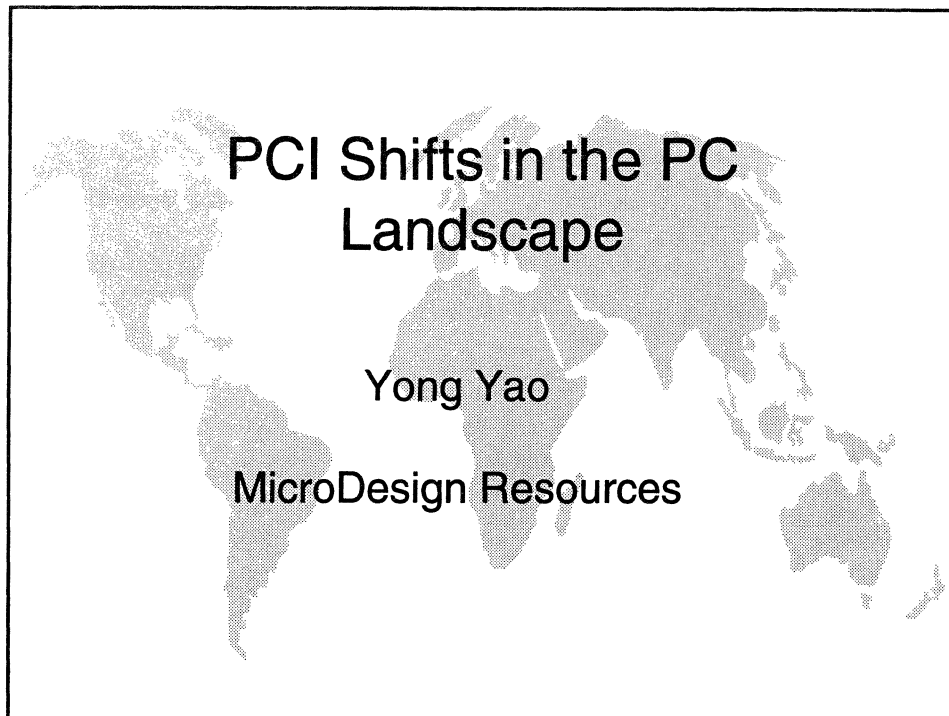
# Thank You!

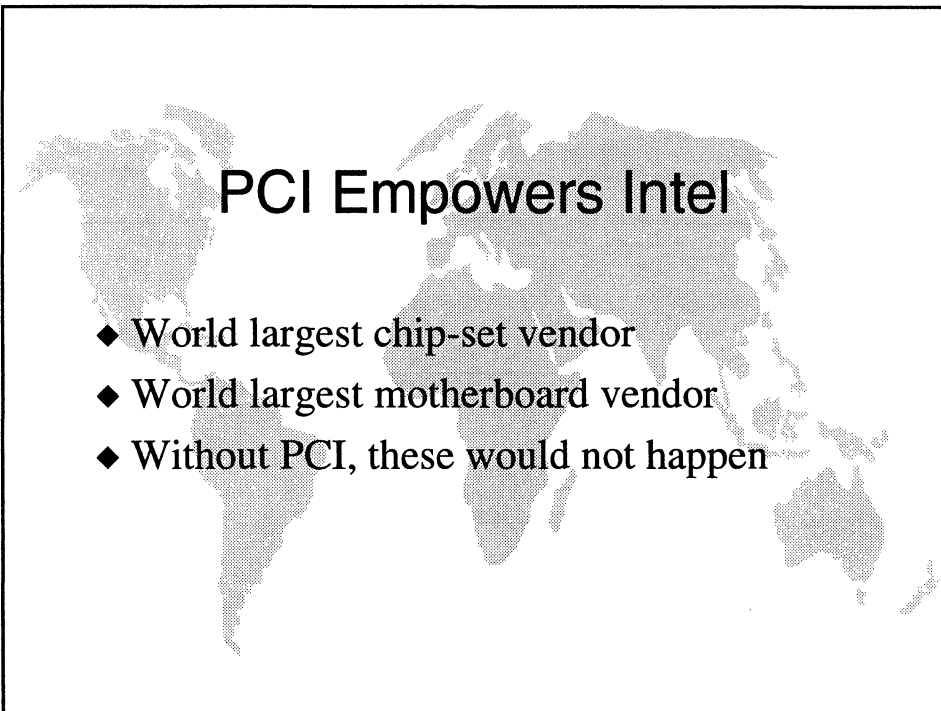
Pro-Log Corporation

Booth #1402

[www.prolog.com](http://www.prolog.com)

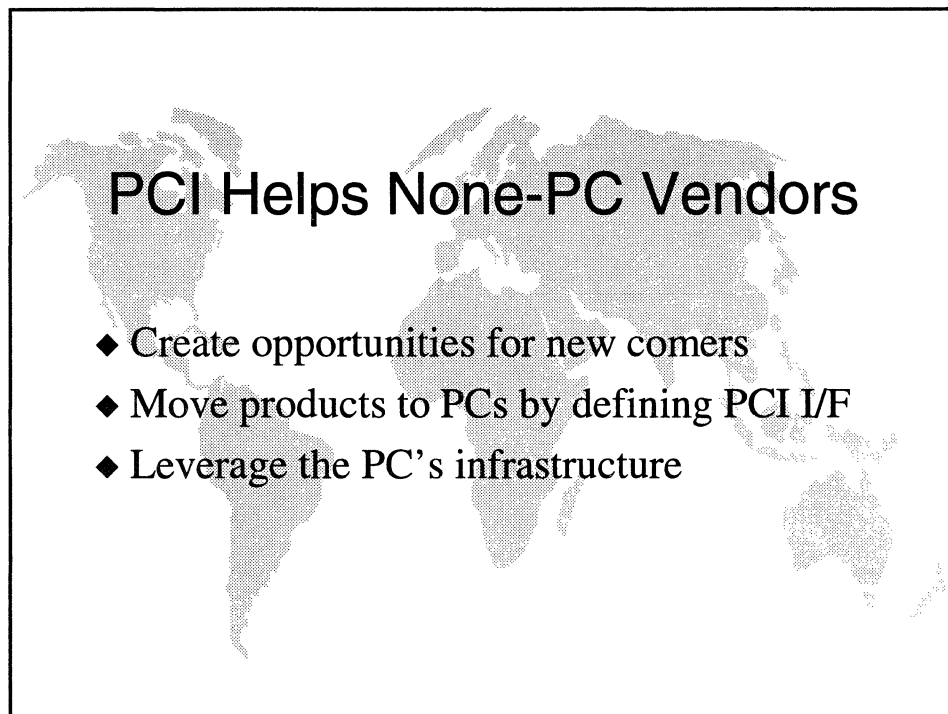
[www.picmg.com](http://www.picmg.com)



A grayscale world map is visible in the background of the slide, showing the continents of North America, South America, Europe, Africa, Asia, and Australia.

## PCI Empowers Intel

- ◆ World largest chip-set vendor
- ◆ World largest motherboard vendor
- ◆ Without PCI, these would not happen

A grayscale world map is visible in the background of the slide, showing the continents of North America, South America, Europe, Africa, Asia, and Australia.

## PCI Helps None-PC Vendors

- ◆ Create opportunities for new comers
- ◆ Move products to PCs by defining PCI I/F
- ◆ Leverage the PC's infrastructure



## PCI Destroys the Old PC Paradigm

- ◆ Make harder for product differentiation
- ◆ Diminish the role of motherboard and core-logic vendors
- ◆ Consolidate the PC market



# The **ATX** Form-Factor

1



## **Presentation Objectives**

- Understand the Goal of Creating ATX
- Understand Industry Forces Driving ATX
  - ◆ Perspective on Motherboard FF Evolution
- Understand ATX Specification Guidelines
  - ◆ Features of ATX
  - ◆ Rear I/O
  - ◆ Mounting Hole Configuration
  - ◆ Power supply
  - ◆ Thermals
- Summary

2



## ATX Goals and Guidelines

### ■ Goals

- ◆ Enable new technologies that other FF's do not support
- ◆ Reduce cost
  - Higher integration on motherboard
  - One system fan

### ■ ATX is an open guideline specification

- ◆ No licence fee
- ◆ Freely available via World Wide Web
- ◆ Mounting hole configurations
- ◆ Component placements
- ◆ Keep out zones

intel

3



## Market Forces Driving ATX

### ■ The PC in the 80's

- ◆ Stand alone business tool
- ◆ High costs
- ◆ Low integration
- ◆ Word processing, databases

### ■ The PC in 1996

- ◆ Consumer product
- ◆ Cost sensitive = High integration
- ◆ Net surfing, video editing, 3D gaming
- ◆ Connectivity/networking crucial
- ◆ New applications drive new I/O

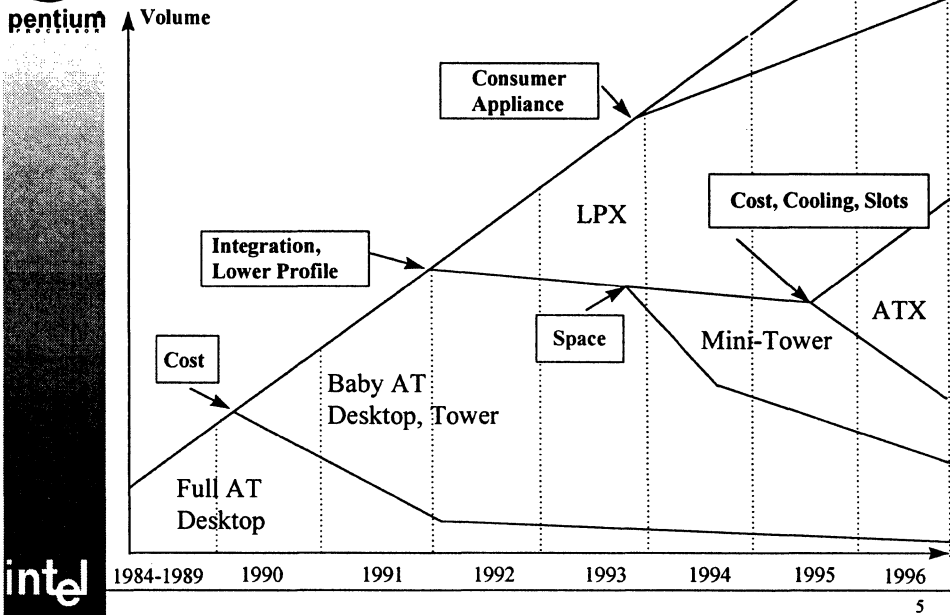
New PC capabilities and needs are driving change

intel

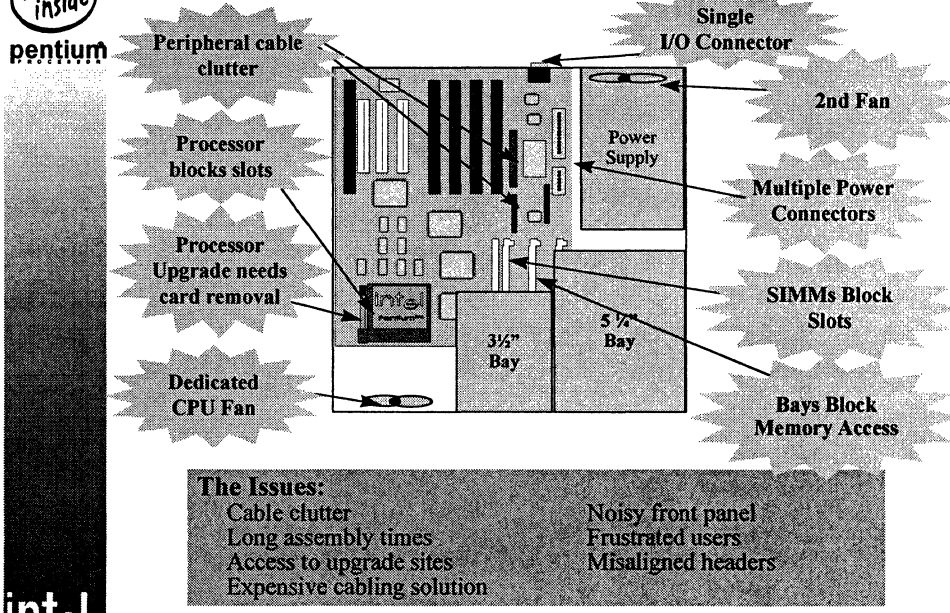
4



# PC Form-Factor Evolution



# Baby AT Design Issues



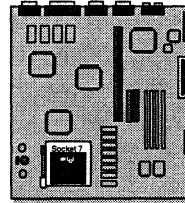




## LPX chassis design

■ LPX is more integrated than BAT, but....

- ◆ Expandability is still lower than desired
- ◆ Riser card adds cost
- ◆ No built in support for multimedia and communications
  - No room for audio, midi/game, video in etc.
- ◆ Second fan required
- ◆ Poor end-user access for upgrades
- ◆ No firm standard - riser dependant on chassis



A new approach is required if we are to meet all requirements



7

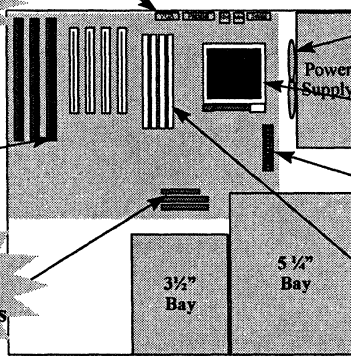


## The ATX Design

Double high expandable I/O

Full length slots

Floppy/IDE connectors close to peripheral bays



Single chassis fan

CPU located near PSU

Single power connector

Easy to access SIMM memory

### Conclusions :

Easy access to peripherals, cables, and memory  
 Reduced noise (single fan)  
 Second fan support if required  
 Faster assembly & configuration

Lower system level cost  
 Improved reliability  
 Supports both Baby AT and ATX form-factors



8



## System Level Savings of ATX



	Baby AT	LPX	ATX
2nd System fan	\$3.00	\$3.00	Not Required
Parallel and Serial Cables	\$4.00	On Board	On Board
Video Riser Card	\$2.50	On Board	On Board
Audio Riser Card	\$5.00	\$5.00	On-board
Mini-Tower Riser for Slots	On Board	\$15.00	On Board
ATX Power Supply (Temporary)	Not Required	Not Required	\$1.50
<b>Total System Adder</b>	<b>\$14.50</b>	<b>\$23.00</b>	<b>\$1.50</b>

Note: ATX can be used for BOTH highly integrated boards and very basic boards.

**ATX Saves Money**



9



## ATX Features

- Improved functionality
  - ◆ Full length slots
  - ◆ Future I/O flexibility
- Improved ease of use
  - ◆ Easy upgrade
  - ◆ Serviceability
- Reduced system level cost
  - ◆ Material cost reductions
  - ◆ Improved manufacturability
  - ◆ Cost efficient cooling



10



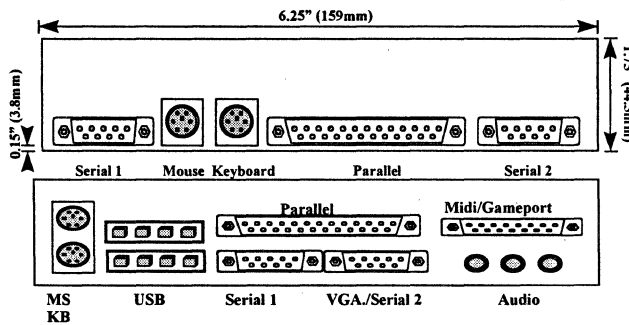
# ATX Guideline Specification Summary

- Rear I/O Panel
- Mounting Holes
- Power Supply
- Thermals
- How to get Information



## Double Height Flexible I/O

- ATX I/O aperture provides more I/O area



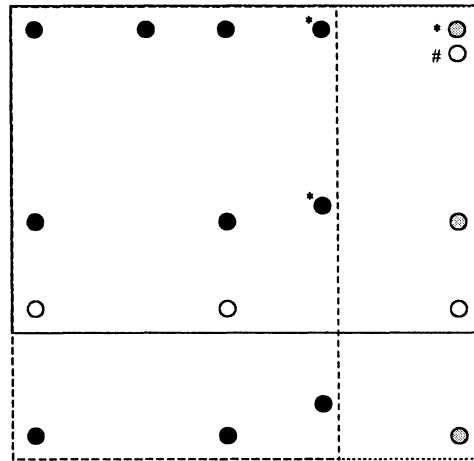
- New I/O needs include
  - ◆ Universal Serial Bus, Video Input, Video Output, TV input, TV output, ISDN, Cable

**Built in flexibility**





# Mounting Holes For Backward Compatibility



Key	
—	ATX
- - -	Baby AT
.....	Full AT
⊙	Full AT hole
●	Baby AT hole
○	ATX hole
*	Hole not used by ATX
#	Optional hole

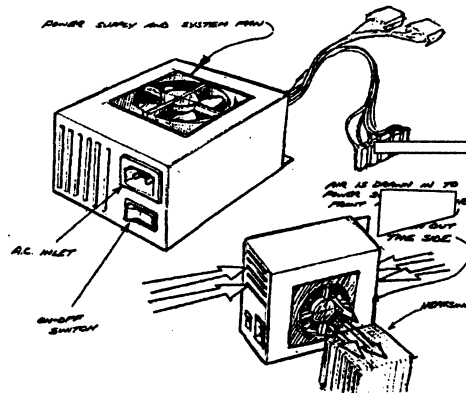


# ATX Power Supply

- An optimised power supply brings even more benefits
- One power connector replaces four
  - ◆ ±5V
  - ◆ ±12V
  - ◆ 3.3V
  - ◆ Soft-power
- Improved manufacturability
  - ◆ installation time
  - ◆ single keyed connector = fewer defects
- Lower cost
- Easier for users and service

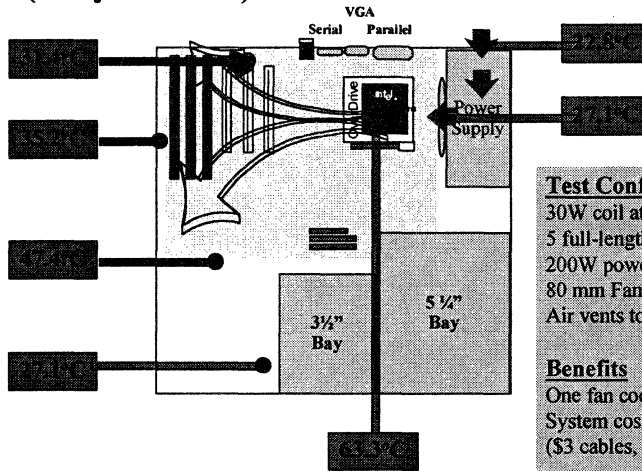
### Example specification

- 28 CFM Forced Cool
- 200 Watts
- 62% Efficiency
- PS/2 Size
- External 92mm Fan
- One 20 Pin Power Connector





## ATX Thermal Test Results (In-System Test)



**Test Configuration**  
30W coil at processor  
5 full-length 10W add in cards  
200W power supply  
80 mm Fan  
Air vents to left side

**Benefits**  
One fan cools system  
System cost reduced by \$11  
(\$3 cables, \$3 fan, \$5 I/O)

**ATX can take the heat!**



## Get the ATX specification on the WWW

- The ATX specification
  - ◆ <http://www.intel.com/pc-supp/motherbd/atx.html>
  - ◆ Available in Microsoft Word for Windows 6.0 and Adobe Acrobat format
- Summary on the PC Platform and What's New Pages
  - ◆ <http://www.intel.com/pc-supp/platform.html>
  - ◆ <http://www.intel.com/pc-supp/whatsnew/index.html>
  - ◆ <http://www.teleport.com/~ATX>





## Summary

- The PC marketplace has changed
  - ◆ Current form-factors not optimised for new needs

- ATX delivers

- ◆ Improved functionality
  - Full length slots
  - Future I/O flexibility
- ◆ Improved ease of use
  - Easy upgrade
  - Serviceability
- ◆ Reduced system level cost
  - Material cost reductions
  - Improved manufacturability
  - Cost efficient cooling

BETTER FEATURES

EASIER TO USE

CHEAPER TO BUILD

- ATX transition is well underway



**BIOS BOOT SELECTION**  
**Frances Cohen**  
**Phoenix Technologies Ltd.**  
**2575 McCabe Way**  
**Irvine, CA 92714**  
**e-mail: frances\_cohen@ptltd.com**

BIOS Boot Selection is a feature which permits increased flexibility of a user's selection of boot devices. It is based on the BIOS Boot Specification which describes a methodology by which the BIOS will identify all IPL (Initial Program Load) devices in the system, prioritize them in the order the user selects, and then sequentially go through each device and attempt to boot. The BIOS must become more intelligent about booting because the PC '95 Specification places additional requirements on the BIOS during the boot process, and there are now more devices that are bootable such as CD-ROM, network remote boot, PCMCIA, etc. It is important that this specification define a boot scheme that is generic and flexible enough to allow booting from virtually any existing IPL device, and for the definition of future IPL devices as well.

The BIOS Boot Specification defines a feature within the BIOS that creates and maintains a list of all the IPL devices found in the system and stores this list in NV memory. IPL devices come in three flavors: BAID (Bios Aware IPL Device), PnP Card, and Legacy. Only BAIDs and PnP Cards are enumerated. Legacy devices are not supported for several reasons. First, they tend to take control of the boot process altogether making them rather unfriendly. Second, they provide no means for identifying themselves as an IPL device. Finally, the BIOS cannot selectively boot from one of several Legacy IPL devices in a system.

The BIOS Boot Specification provides one basic feature, the IPL Priority. The IPL Priority is a user-specified priority of IPL devices that is arranged in Setup. This boot order is similar to the common feature of boot A: then C: or vice versa, but supports additional IPL devices. Also, the number of IPL devices in the system may vary from one power-on to another. Each time the user turns on the system all IPL devices in the system are enumerated.

Additionally, the BIOS Boot Specification defines the BCV (Boot Connection Vector) Priority. The BCV Priority is a user-specified priority list of INT 13h Device Controllers that is arranged in Setup. This list specifies the order that the controllers will be called to install their INT 13h drive support during POST.

If an IPL device fails to load an O/S, the BIOS regains control and attempts to boot from the next available IPL device. This procedure will continue until all possible IPL devices have been exhausted. Only then will the BIOS display a message that an O/S cannot be found, wait for a key stroke, and then invoke INT 19h again. This method ensures that the BIOS has intelligently made every attempt to boot.

The BIOS Boot Specification encompasses the boot process of both PnP and non-PnP systems. A standard AT compatible system (also called a Legacy system) is much simpler than one with a PnP BIOS because it only supports BAIDs. A Legacy system does not need to provide any dynamic IPL device enumeration or configuration, nor does it support PnP Cards in their native mode. This is because the number of IPL devices in such a system will never change.

\* Excerpt from the BIOS Boot Specification Version 1.00, October 11, 1995



**Notebook Docking:  
Techniques and Considerations**

Jim Kelsey, Technologist  
SystemSoft Corporation  
2 Vision Drive  
Natick, MA 01760  
(508) 651-0088  
jkelsey@systemsoft.com

January 12, 1996

*(excerpts from article published in Personal Engineering, January, 1996)*

## **Introduction**

The debut of Windows '95 and its Plug and Play capabilities has spurred a new interest in dockable PC's, that is notebooks that connect to a convenience base to become a super-notebook or desktop equivalent system. This article talks about the styles, benefits and pitfalls of buying into "Desktop To Go".

While docking station technology is by no means new (in its simplest form, the docking station is simply an extension of the portable's internal ISA bus), it's capabilities have been redefined by Windows '95 to include hot docking, in which case the notebook and dock station can connect while the operating system is active, without requiring the user to re-boot or cycle power. During a hot dock, the Plug and Play operating system (Windows '95 for now) and system firmware work together to detect, enable and configure the newly arrived peripherals in the docking station.

As you might expect, docking station systems come in all sizes, shapes, capabilities and price ranges. Some vendors even stretch the capabilities of Windows '95 by offering a choice of docking stations that connect to single notebook. This allows users to plug in to a multi-drive, high-resolution video, networked power-user station at the office, or a scaled-back home dock that contains a sound blaster and external connectors for a favorite external keyboard or mouse.

### **Styles of docking stations.**

Docking stations come in one of three styles -- the mini-dock, the port replicator and the convenience base.

The simplest of the three, the mini-dock, allows users to connect to a single, or small handful of peripherals. For example, the DEC HiNote machine can be outfitted with floppy drive/PCMCIA mini-dock or a more sophisticated multi-media mini-dock that provides a CD-ROM drive as well as sound capabilities. Mini-docks allow users to tailor their portable system to the chore at hand; a quick business trip across town might involve simple data entry, in which case the larger multimedia mini-dock can stay back at the office. A graphic business presentation, on the other hand, might require multi-media capabilities, so at the expense of size and weight the user would pack both the notebook unit and the multimedia mini-dock.

## Docking Stations -- A Comparison

	Dock Support	Bus Type	Slots	Portability	Peripherals
Mini-Dock	Hot Warm Cold	VL ISA PCMCIA Proprietar y	No	Good	CD-ROM, PCMCIA, floppy
Port Replicator	Hot Warm Cold	VL ISA PCI	No	Fair to Good	Serial/Parallel, IR Port, External Keyboard, External Mouse, External Video, etc.
Convenience Base	Hot Warm Cold	VL ISA PCI	Maybe	Poor	Additional Fixed Disks, Floppy Disks, Battery Charger, CD-ROM, Network, Plug-In Slots, etc.

Next in terms of price/performance, port replicators generally serve two purposes. First, they act as a semi-permanent base for the more cumbersome, desktop peripherals such as full-size keyboards, external mice and monitors. Second, port-replicators often introduce new functions, such as an infrared port or network connector to the basic notebook system, enabling it to become more like a desktop, although without adding any mass storage or plug-in adapter slots. As the name suggests, a port replicator simply replicates the plug-in ports located at the rear of the machine by routing the internal VL/ISA or PCI bus to duplicate connectors in the replicator unit. High-end port replicators might provide additional serial or parallel ports not available on the notebook due to size constraints caused by the connectors or the notebook chassis.

At the top of the docking station food chain sits the convenience base. This type of unit is designed to fully transform the portable system into a true, desktop unit. Because of its size, weight and reliance on wall-power, it rarely, if ever leaves the top of the desk. Convenience bases generally come in one of two flavors -- those with slots and those without. Units without slots are not necessarily less elegant. As I'll mention later, designers of this type of unit often wish to forego the complexity introduced by opening the system's architecture up to plug-in boards.

### Docking Methods

Docking capable systems support up to three dock modes -- cold docking, warm docking, and the newly possible Windows '95 style hot dock.

All docking-capable systems support cold-docking, in which case power is removed from the system when the notebook is either inserted in, or removed from its counterpart mini-

dock, port replicator or convenience base hardware. The simplest docking systems support cold docking only.

Units that support warm docking can be inserted into, or removed from their docking station either when totally off, or when in low-power suspend mode. Since the docking station is often powered by wall-current, many units disable power-savings modes and automatically power on or resume when the system is fully docked. Since power-saving are disabled in a docked system, the notebook unit cannot suspend and a warm-undock is not possible. Additionally, if the user resumes a newly docked notebook, he or she is effectively completing a hot dock. Because the circuitry required to transform warm-dock systems to hot-dock systems is relatively simple, many IHV's absorb the additional cost and complexity to achieve true hot docking. As a result, the number of true, warm-dock-only systems is quite small.

Hot-docking capable systems allow the user to insert the notebook system into its docking station when the unit is powered off, in low-power suspend mode, or fully powered and executing within the operating system. Hot docking requires support from the Plug and Play operating system (such as Windows '95). During its boot sequence, the Plug and Play operating system broadcasts its presence to the system firmware. Since the system firmware (Plug and Play BIOS) is aware of the Plug and Play operating system (or lack thereof) many systems will intentionally reboot if the user attempts to hot-dock while running a non Plug and Play OS.

#### **Physical docking mechanisms:**

In terms of the physical mechanism used to insert and/or eject a notebook from its dock, there are the following three variations, listed in the order of their ability to prevent data loss.

**Surprise Style** (Least Expensive, Least Effective at Preventing Data Loss) A user of this type of system can insert the notebook in, or remove the notebook from its docking station at any time.

**Honor System** (More Expensive, Medium Effectiveness at Preventing Data Loss) A user of this type of system must activate a switch or OS applet to inform the operating system that the notebook is about to be removed from its docking station during an undock. Once the OS has determined that undocking is safe, a message appears on the screen signaling to the user that the notebook can now be removed.

**VCR-Style** (Most Expensive, Highest Effectiveness at Preventing Data Loss) A user of this system activates an OS applet or system switch to indicate that he/she wishes to undock the notebook from its convenience base. Upon receipt of a special message from the Plug and Play operating system signaling that it is safe to undock, the system firmware ejects the notebook via a VCR-like mechanically locking servo motor apparatus.

## **The Dock/Undock Sequences.**

Users and designers alike must take care to handle the somewhat delicate process of inserting or removing a unit from its docking station. In a cold-dock only system, both the notebook and the docking station must be powered off before a dock or undock is allowed to occur. As far as warm-docking is concerned, the number of true warm dock/undock systems available is too small to be worth discussing. This leaves the hot-dock capable system, whose capabilities present a plethora of challenges not only to the designers of the system, but to the architects of Plug and Play operating systems who must deal with the almost instantaneous addition or removal of peripherals and mass storage devices in the system.

Consider that a fully docked notebook system might have files open in a number of places -- on a docking station-based fixed disk, network connection or even a PC (PCMCIA) SRAM card. Prior to any physical undock, the Plug and Play operating system must take care to flush and close all open data files and remove the docking station-based mass storage device (and other peripherals) from its registry, which maintains a catalog of all currently attached peripherals.

The OS' task of managing an undocked notebook is equally daunting if you consider that at any time, the user might dock to any of a number of dissimilar docking stations, each of which contains new and additional devices, each of which requires initialization and system resources, such as IRQ channels, DMA channels and I/O space.

The key to successful docking and undocking depends on close coordination between the system's hardware, firmware and Plug and Play operating system. The following section details the exact sequence that occurs between a Plug and Play BIOS-equipped dockable notebook and Windows '95. Remember that a dock or undock generates an SMI (System Management Interrupt) that allows the system firmware to execute regardless of the foreground mode of the OS and its applications.

### **The Docking Sequence**

During a dock sequence, the system firmware and Plug and Play operating system are responsible for locating, enabling and configuring (assigning resources to) any peripherals in the docking station. The Plug and Play operating system and system firmware share the responsibility of locating, configuring (assigning resources to) and enabling docking station devices. The docking sequence proceeds like this:

**Step #1:** The user docks his or her portable system to its port replicator, mini-dock or convenience base. This generates an SMI (System Management Interrupt) that transfers control to the system's Plug and Play firmware. Interestingly, all dock events are treated as if the system were a Surprise Style docking station, as described above. The more elaborate Honor System and VCR-style hardware is used only during an undock to prevent data loss.

Step #2: The portable system identifies the docking station, configures any slot-based PCI devices (Plug and Play ISA devices are handled completely by the OS) and prepares a DOCKING\_STATION\_INFO structure for the operating system.

Step #3: The system firmware activates any VCR-style or other locking hardware to insert the notebook into its dock and broadcasts the message DOCK\_CHANGED to the operating system by setting bit 0 in the Plug and Play BIOS event flag and supplying the Plug and Play BIOS event handler with the DOCK\_CHANGED message.

Step #4: The Plug and Play operating system (which has polled bit 0 of the event flag and found it set) invokes the Plug and Play BIOS GetEvent function to find that the system has docked. The Plug and Play BIOS clears bit 0 of the event flag during the GetEvent function.

Step #5: The Plug and Play operating system invokes the Plug and Play BIOS GetDockingStationInfo function to find out the type of docking station to which the notebook has been docked, re-enumerates the system, arbitrates the system's resource (IRQ, DMA channel, I/O range and address range) usage, notifies its drivers of any resource changes and resumes foreground execution.

## **Peripheral Support**

Docking station peripherals belong to one of two classes -- internal and slot-based. Internal peripherals are those managed by the system's Plug and Play BIOS. Slot-based peripherals are those that can be identified by the Plug and Play operating system, but might need system firmware support during the dock. The following list shows the rules involved in docking:

No late-arriving ROMs. Neither the system firmware nor the Plug and Play operating system will invoke an expansion ROM that belongs to a device in the docking station. This is because the real-mode address space in which the expansion ROMs normally exist has already been spoken for by virtual mode RAM UMBs (Upper Memory Blocks) PCMCIA controller memory windows or other memory mapped devices. Although the Plug and Play OS has a record of how expansion ROM space has been allocated, it is unable to deal with the specifics of the system's PCI controller shadow RAM, ROM chip select regions, etc. Such devices must be configured by OS level device drivers.

No OS Support For PCI Devices Behind A PCI Bridge. The Plug and Play operating system can enable (via the PCI configuration space command register) but not configure PCI devices that reside behind a PCI-PCI bridge. Therefore, if the system firmware does not configure such PCI devices, they remain unusable after a dock has completed.

No hot docking support for late-arriving video controllers. Generally, the Plug and Play operating system is unable to switch video adapters during a dock. For instance, the video

card in the docking station might not support the video mode in which the portable system is currently executing prior to the dock.

No hot docking in a non-Plug and Play operating system. Unless the operating system can handle the messaging that must occur between the OS, its drivers and file system and the system firmware when a dock occurs, hot docking might result in data loss or system crashes. Currently, Windows '95 is the only operating system that completely supports hot docking. Support for hot docking is expected to arrive next year in other operating systems, such as OS/2 and Windows/NT.

### **The Undocking Sequence**

The undocking sequence is usually more critical because it implies “pulling the plug” on devices that are currently playing. The following list details the communication that occurs between the system firmware and Plug and Play operating system during an undock operation. NOTE: This list applies only to Honor-System or VCR-style docking stations as described previously. If a user removes a Surprise Style docking station, the OS' only job is to try and recover from the undock with minimal data loss.

**Step #1:** The user signals an undock via an OS applet or physical switch on the portable or docking station unit.

**Step #2:** Having detected that the user is about to remove the portable from its docking station, the Plug and Play BIOS sets its event flag and posts the message ABOUT\_TO\_CHANGE\_CONFIG.

**Step #3:** The Plug and Play operating system, which constantly polls the Plug and Play BIOS event flag, detects a Plug and Play BIOS event and invokes the BIOS GetEvent function. It receives the message ABOUT\_TO\_CHANGE\_CONFIG. During the GetEvent call, the Plug and Play BIOS clears its event flag so the event will not accidentally be detected twice.

**Step #4:** The Plug and Play operating system determines if it is safe for the user to undock. If so, it closes any files residing on docking station-based file systems, such as PC Cards, network adapters, etc and it invokes the Plug and Play BIOS SendMessage function with the OK message. Otherwise, it invokes the SendMessage function with the ABORT message. If the OS aborts the undock, the notebook system remains attached to its docking station and the user is free to try undocking again at a later time.

**Step #5:** Upon receipt of the OK message, the Plug and Play BIOS ejects the portable PC from its docking station if necessary. It then broadcasts the message DOCK\_CHANGED to the operating system via its event flag/GetEvent messaging mechanism.

Step #6: The operating system re-enumerates and reconfigures the portable system's devices and continues execution in the foreground.

### **So, What's Hot and What's Not???**

So far, it seems that docking is pretty straightforward, provided that the OS/system coordination shown above occurs in the proper order and the engineers who have implemented this support code stayed up late ensuring that everything works as advertised.

Hot docking is hot for users that frequently switch between docked and undocked state. Rather than having to repeatedly sit through system boot sequences, these users are quickly back to work after the OS completes its re-enumeration and reconfiguration process.

Hot docking is also hot for users that have ISA cards plugged into their docking station. Despite its Plug and Play capabilities, Windows '95 relies heavily on its ability to snoop for and detect the current configuration of legacy ISA plug-in adapters. Fortunately for the user, Windows '95 built in legacy ISA adapter support is both robust and accurate.

Hot docking is not for the user of a non-Plug and Play operating system. If you hot dock in normal, bare-bones DOS, don't expect peripherals in the docking station to magically start working -- the support simply doesn't exist. For this type of user, however, the boot process is probably much faster than Windows '95 so the process of cold booting should be less painful.

Hot docking is also not for the user who's constantly swapping ISA or PCI cards in and out of the docking station. Windows '95's detection algorithms are sophisticated, but if you keep adding and removing hardware, they're not exactly fast and the benefits of the speed of hot docking are sometimes nullified by the time Windows '95 needs to figure out what you've added and removed from the system.

For more information on this topic, the Plug and Play BIOS Specification version 1.0A is available on the CompuServe Plug and Play forum by logging into CompuServe and typing: GO PLUGPLAY.



## Multimedia Roundtable

Multimedia applications are the most demanding clients of the PCI bus. Video, 3D, and audio data streams consume large portions of the PCI bandwidth while imposing strict latency requirements. Interactive multimedia applications must attain a high frame rate in order to be compelling. Once they have achieved a high frame rate, the focus changes to increasing quality. Quality is improved by increasing the amount of detail and thus the amount of data that needs to be transferred. Thus, the relentless press for the highest level of realism stresses PCI bridge chips to their limit. Devices optimized for CPU-to-Memory accesses at the expense of CPU-to-PCI or PCI-to-Memory accesses will suffer. The challenge for chipset designers is to balance the requirements of spreadsheets with those of interactive multimedia. Future PCI bus utilization indicates a trend towards bus mastering multimedia devices streaming ever increasing volumes of 3D textures, audio, video, and communications data.

# **Bridging the PCI to a Secondary Multimedia Bus Can We Plug and Play?**

*Larry Chisvin, S3 Incorporated*

A trend that has already started and is expected to accelerate over the next several years is the use of a secondary dedicated bus to handle multimedia devices such as MPEG decoders, audio devices, and videoconferencing solutions. These devices are often attached directly to the graphics device through a side port, and the data for the device is routed to and from the PCI bus using the graphics chip as a bridge. One major problem faced by this type of configuration is how to handle the Plug and Play function when both multiple devices and multiple bus operating nodes need to be supported. This presentation discusses the problem briefly, explaining why it exists and what the obstacles are, then suggests some general solutions.

# **CAD Tools**

*Jim Lipman, EDN*

The proliferation of the PCI standard has spawned a number of companies offering both hardware and software products for the designer of systems using this interface. Hardware products for ASIC-based systems include pre-designed and pre-verified synthesizable cores that you can embed into your chips. Software tools encompass those used to design PCI cores as well as those needed for designing the chips that use them. The CAD category also includes software tools required to measure PCI-based system performance.

In the CAD Tools session, you will hear papers from companies at the forefront of available CAD tools for the development, verification, and optimization of PCI-based designs.

**David L. Evans**  
Vice President, Strategic Marketing

## *Agenda*

- ◆ **Technical Data Freeway**
- ◆ **Customer Profile**
- ◆ **Customer Issues**
- ◆ **PCI Core**
- ◆ **Designing with the PCI Core**



*Company*

- ◆ **Founded in 1992**
- ◆ **500 Plus Years of Complex ASIC Design Experience**
- ◆ **Over 100 years of Sales, Marketing Experience**

Technical Data Freeway 1996

DLE-3



*Company*

### **Mission Statement**

Technical Data Freeway is in the business of providing process independent products and services to companies that develop complex, integrated ASIC systems with compressed development schedules.

Technical Data Freeway 1996

DLE-4



## *Customer Profile*

Technical Data Freeway's objective is to be a strategic *Partner* to high-technology companies with fast moving product development cycles

to:

- ◆ Reduce Risk
- ◆ Shorten Time to Market
- ◆ Preserve Alternatives
- ◆ Cost Effective

Technical Data Freeway 1996

DLE-6



## *Customer Profile*

### **Market Trends**

- ◆ Increased Global Competition
- ◆ Shrinking Product Life Cycles
- ◆ Increasing Product Complexity
- ◆ Varied Technology Choices
- ◆ Faster Clock and Data Rates
- ◆ Limited, Changing and Uncertain Fab Capacity



Technical Data Freeway 1996

DLE-6



## Customer Issues

- ◆ Late To Market
- ◆ Increase Design Productivity
- ◆ Increase Design Complexity
- ◆ Reduce Design Cycle Time
- ◆ Technology Independence
- ◆ Preserve CAE/CAD Investment

Technical Data Freeway 1996

DLE-7



## Customer Issues

### Late to Market Issues

- ◆ Miss Market Leader Margins
- ◆ Lengthen time to recover R&D
- ◆ Playing Catch-up



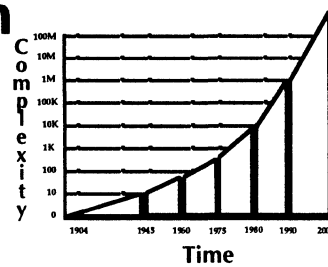
Product Life (months)	18	18	18
Total Expected Revenue	\$25 Mil.	\$50 Mil.	\$100Mil
1 Day	\$137,000	\$274,000	\$547,000
1 Week	\$953,000	\$2 Mil.	\$3.8 Mil.
1 Month	\$4 Mil.	\$8 Mil.	\$16 Mil.

Technical Data Freeway 1996

DLE-8

## What is good Design Productivity?

- ◆ 1965 - One Transistor per Day
- ◆ 1985 - 10 Gates Per Day
- ◆ 1995 - 100 Gates Per Day
- ◆ 2000 - 25,000 Gates Per Day



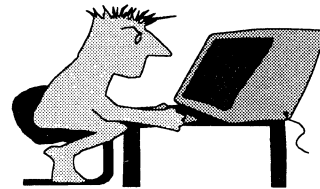
In two years we will be able to routinely fabricate IC's with over 2 million gate but they will take longer than 10 man years to design.

In four years we can fabricate 25 million gate IC's

Technical Data Freeway 1996

DLE-9

## What is good Design Productivity?



### 1995 AVERAGE ASIC DESIGN

- ◆ 30,000 Gates
- ◆ Design Cycle Time 45 Weeks

### 5 YEARS AVERAGE ASIC DESIGN

- ◆ 300,000 Gates
- ◆ Design Cycle Time 25 Weeks

Technical Data Freeway 1996

DLE-10



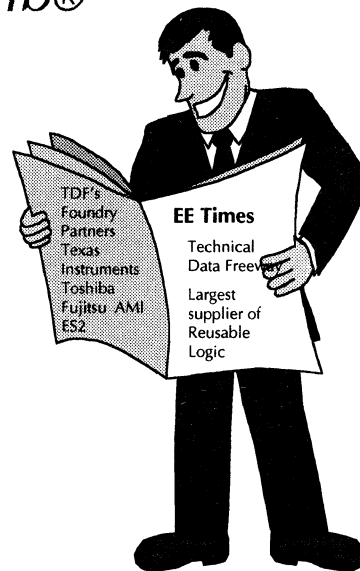
## How to Improve Productivity?

Relative Effect

- |                         |             |
|-------------------------|-------------|
| ◆ Good People           | Low         |
| ◆ Good Tools            | Medium      |
| ◆ Good Methods          | Medium      |
| ◆ <b>Reusable Logic</b> | <b>High</b> |

## Deliverables

- ◆ RT Level Source Code
- ◆ Synthesis Script
- ◆ Documentation
- ◆ Test Bench
- ◆ Training

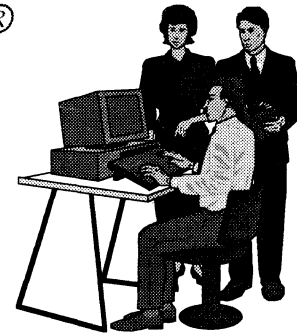


<p><b>TELECOM</b></p> <p>SONET-Generator SAR 622/155 SONET Frammer 10/100 Ethernet VITERBI Reed-Soloman</p>	<p><b>FAX/MODEM</b></p> <p>320C25 Equivalent A/D D/A Fax/Modem Functions Function Software v.17, v29, v32, v.32bis</p>	<p><b>MULTIMEDIA</b></p> <p>VGA MPEG1 MPEG2 NTSC Encoder* Audio MPEG1</p>	<p><b>MICROPROCESSORS CONTROLLERS</b></p> <table> <tr><td>6502</td><td>68HC11</td></tr> <tr><td>6805</td><td>8086</td></tr> <tr><td>8051</td><td>8088</td></tr> <tr><td>Z80</td><td>80186</td></tr> <tr><td>8031/32</td><td>80188</td></tr> <tr><td>8031 Turbo</td><td>R-3000</td></tr> </table>	6502	68HC11	6805	8086	8051	8088	Z80	80186	8031/32	80188	8031 Turbo	R-3000
6502	68HC11														
6805	8086														
8051	8088														
Z80	80186														
8031/32	80188														
8031 Turbo	R-3000														
<p><b>BUS LIBRARY</b></p> <p>PCI 32 PCI 64* PCMCIA IIC* FIREWIRE* CARD BUS*</p>	<p><b>FUNCTION LIBRARY</b></p> <table> <tr><td>INTERRUPT</td><td>ALU, etc..</td></tr> <tr><td>BASIC I/O</td><td>RT CLOCK</td></tr> <tr><td>UART</td><td>INPUT/OUTPUT</td></tr> <tr><td>TIMER</td><td>UART - 16450</td></tr> <tr><td>DMA</td><td>UART - 16550</td></tr> </table>	INTERRUPT	ALU, etc..	BASIC I/O	RT CLOCK	UART	INPUT/OUTPUT	TIMER	UART - 16450	DMA	UART - 16550	<p><b>DSP FUNCTION LIBRARY</b></p> <p>MATH FUNCTIONS FIR FUNCTIONS IRR FUNCTIONS OTHER FILTERS VITERBI</p>			
INTERRUPT	ALU, etc..														
BASIC I/O	RT CLOCK														
UART	INPUT/OUTPUT														
TIMER	UART - 16450														
DMA	UART - 16550														

\* Under Development

## Quality Support

- ◆ Knowledge Transfer
- ◆ Direct Access to the Developers
- ◆ Architectural Options
- ◆ Implementation
- ◆ Custom Development



## Product Development Organization

◆ TELECOM	25 Engineers
◆ MULTIMEDIA	25 Engineers
◆ BUS LIBRARY	15 Engineers
◆ MICROPROCESSORS	15 Engineers
◆ DSP FUNCTION LIBRARY	15 Engineers
◆ FAX/MODEM	10 Engineers
◆ BASIC FUNCTIONS LIBRARY	10 Engineers
◆ R&D	10 Engineers
TOTAL	125 Engineers

Technical Data Freeway 1996

DLE-15

## Product Development Program

- ◆ Market Research
- ◆ Product Specification
- ◆ Create Test Bench
- ◆ Create Synthesizable code and Script
- ◆ Floor planning & Place/Route
- ◆ Verify Against Specifications
- ◆ Evaluation Boards/Software Drivers Development
- ◆ Complete Documentation

Technical Data Freeway 1996

DLE-16



Core\_Lib®



## BUILD

VS

## BUY

Late to Market  
Have or Develop Expertise  
Developmental Risk  
Partial Completeness  
Partial Documentation  
Next Generation  
Exceed Budget

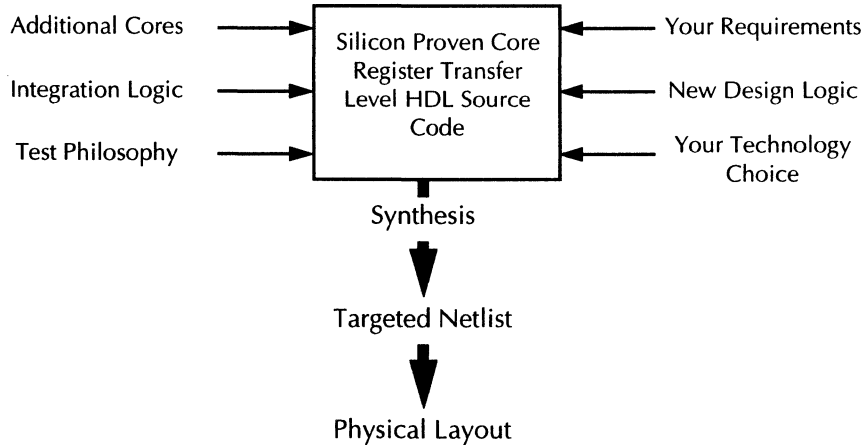
Ready Now  
Center of Excellence  
Silicon Proven  
Complete Coverage  
Complete Documentation  
Migration Path  
Known Cost

Technical Data Freeway 1996

DLE-17



## Core Methodology



Technical Data Freeway 1996

DLE-18



## TOUCAN TECHNOLOGY OVERVIEW

- ◆ Toucan Technology is an company specialising in the design of
  - ◆ Electronic Systems
  - ◆ ASICs
  - ◆ Synthesisable cores with custom features
- ◆ Satisfied customers to date include Hewlett-Packard, Apple, Digital UK and other multinational clients.
- ◆ Proven systems expertise in Telecommunications, Data Networking and Computer Bus Technology.
- ◆ Staff have worked on 30+ ASIC projects with 11 different ASIC vendors.
- ◆ Proven ISO9001 based design process.

Technical Data Freeway 1996

DLE-19



## PCI CORE PROJECT OVERVIEW / GOALS

- ◆ High performance (132 MB/Sec)/ Zero Wait State operation
- ◆ Provide a guaranteed data delivery service to application
  - ◆ Core maintains transaction context (address and data)
  - ◆ Application only has to re-initiate transaction
- ◆ Customisable Functionality
- ◆ Full compliance to Rev 2.1 PCI Specifications
- ◆ Simple generic application interface
  - ◆ Request / Ready handshake protocol
  - ◆ Separate Target and Initiator busses
- ◆ Single clock, synchronous design supporting a scan based test methodology
- ◆ Technology independent

Technical Data Freeway 1996

DLE-20



## **PCI CORE DESIGN METHODOLOGY VHDL AND VERILOG**

- ◆ PCI CORE (without FIFO block) consists of 4,200 lines of synthesisable VHDL/Verilog Code
- ◆ PCI CORE with guaranteed data delivery synthesises to:
  - ◆ 6,500 gates in TI TGC2000 (0.65um) Gate Array technology
  - ◆ 7,000 gates in TSMC (0.65um) Std Cell technology
  - ◆ 9,000 gates in NEC CMOS 8 (0.65um) Gate Array technology
- ◆ FIFO block (16 x 32) synthesises to 3,000 gates in NEC CMOS 8
- ◆ All code is IEEE 1076-1993/Verilog HDL compliant
- ◆ Coding style is based on internal Toucan procedure which provides guidelines for hardware design using VHDL/Verilog
  - ◆ Extensive code commenting
  - ◆ Consistent code structure

Technical Data Freeway 1996

DLE-21

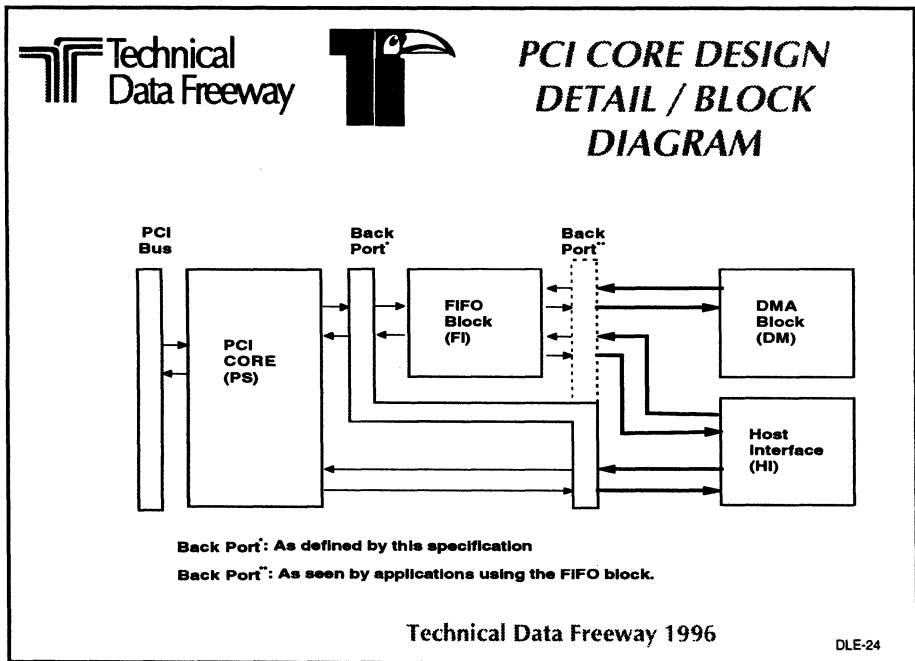
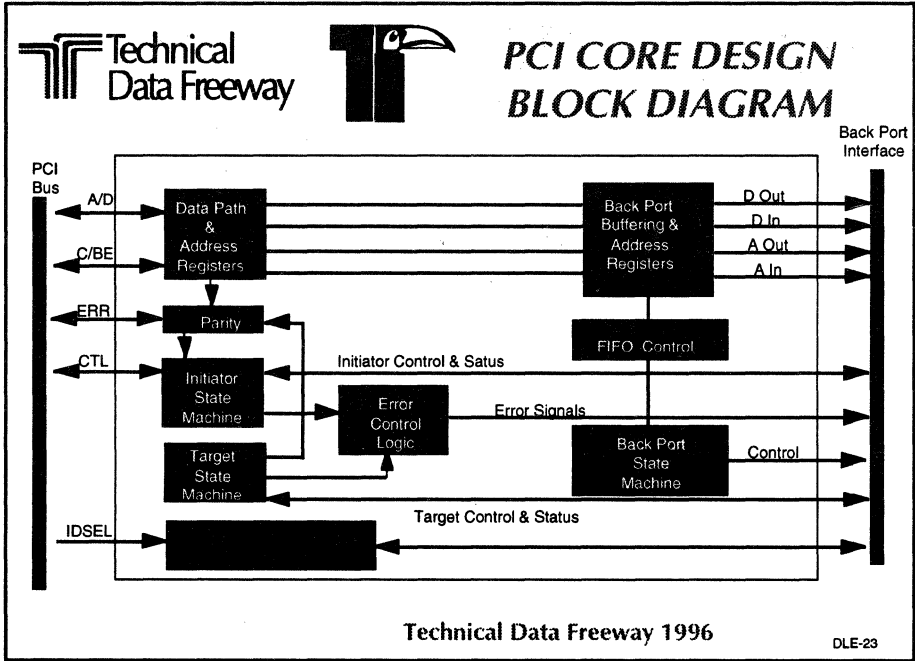


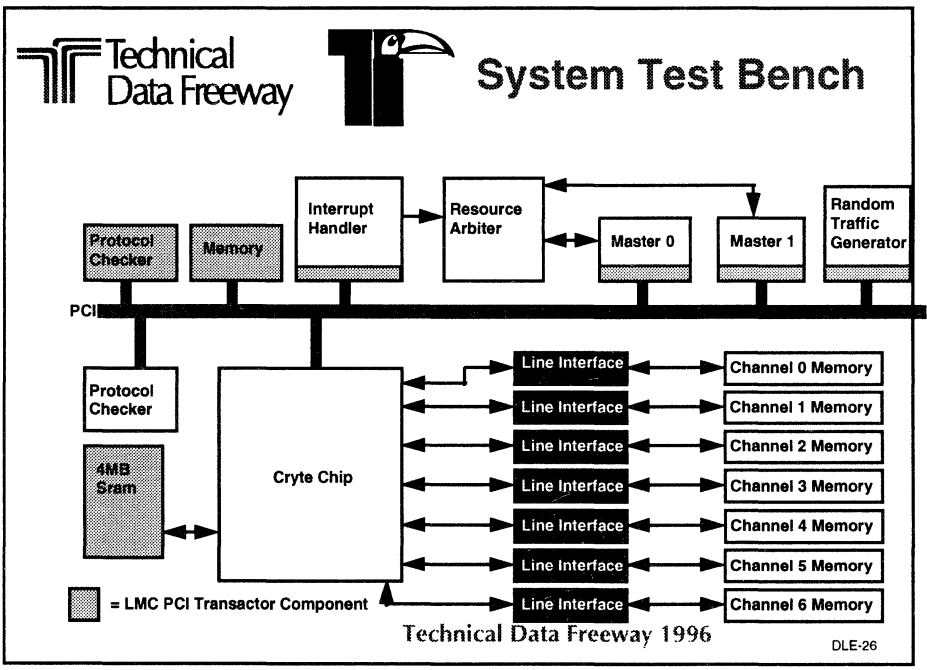
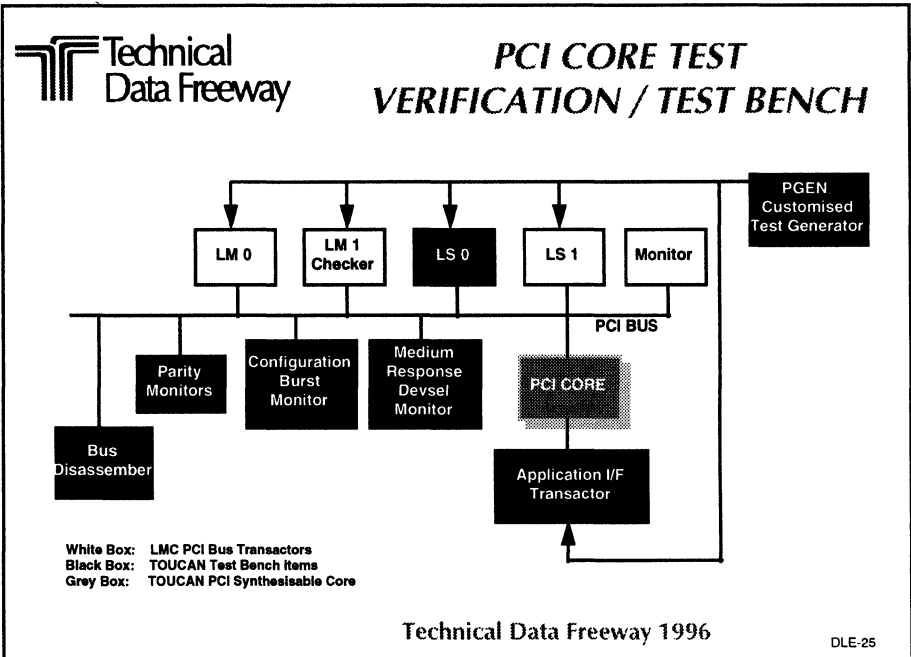
## **PCI CORE DESIGN DETAIL / DESIGN GOALS**

- ◆ Provide simple backport design with high performance features.
  - ◆ Data buffering used to balance the data flow between the application and the PCI Bus.
  - ◆ Design for a direct or FIFO based interface.
  - ◆ Simple handshaking protocol (Bus-Req, Skt-Rdy, BP-Rdy, Last-Data, D-Beat).
  - ◆ Designed for zero wait state initiator and target performance.
  - ◆ Separate initiator and target control and data paths.
- ◆ Provide full error condition handling while meeting the goal of providing a guaranteed data delivery service.

Technical Data Freeway 1996

DLE-22









## *PCI Core Summary*

- ◆ PCI Core Architected for Maximum Continuous Throughput
- ◆ Designed for Fast Synthesis
- ◆ Well Documented
- ◆ User Interface is Easy to Use, Efficient and Flexible
- ◆ 2nd Customer from Purchase to Tapeout in 6 Weeks with First Silicon Success

Technical Data Freeway 1996

DLE-27

## THE PROBLEM OF MODEL AVAILABILITY FOR SIMULATION OF DEVICES AND SYSTEMS

Dave Apte

Omniview, Inc.

100 HighTower Blvd., Suite 201

Pittsburgh, PA 15235

Phone: (412) 788 9492

Fax: (412) 788 0308

e-mail: apte@omnivw.com

### ABSTRACT

*The complexity and speed of devices and systems is ever increasing, and so is the pressure of time-to-market. To design a system which works the first time without costly redesign cycles, it is becoming necessary to use simulation in the design flow. This becomes even more critical when designing with devices that are not currently available for prototyping. However, not many people use simulation because of the lack of good simulation models, especially of the latest devices. Expertise in HDLs is also scarce, which makes it difficult to create necessary models in-house. ALCHEMIST® is a tool that can be used by both semiconductor vendors and system designers to solve this problem. ALCHEMIST creates VHDL and Verilog® source models of devices using graphical input. No expertise in an HDL is needed. ALCHEMIST can be used by semiconductor vendors to distribute simulation models of their latest (and even future) devices. System designers can use it to create the models as they need them. It is easier to upgrade and change models as necessary, since the users have control over the model. The models include accurate timing information, thus making it easy to detect problems with timing violations in the system before prototyping.*

### INTRODUCTION

The complexity of ICs and systems is ever increasing. At the same time, time-to-market pressures are reducing design time, and increasing pressure on designers to “get it right” the first time. Designers no longer have the luxury of multiple design cycles, building prototypes and verifying the design each time. They may not even have all the ICs in the system available to prototype. Under such circumstances, simulation of the design becomes a very important part of the design flow. Full simulation of the system is the only way to verify that all the devices in the system (whether ASICs or off-the-shelf components) work as designed, and also work together as a system. Physical prototypes do provide full functional verification, but cannot provide min and max timing verification, which is critical in high speed designs. Static timing verifiers and other such tools provide a degree of verification of the design, but it is necessary to do a full simulation to explore all the different interactions between devices of the system and verify them.

Simulation, however, brings its own set of problems. Simulation models for the latest devices are not available or are expensive. Writing your own models requires VHDL/Verilog expertise. Even if

you are an expert in an HDL, you may need to distribute the model to other designers, which means that the model must be effectively documented for proper use of the model. VHDL or Verilog source code is not really good documentation, since the other designers will also need to be HDL experts. Separate documentation (say, in the form of descriptions/diagrams, etc.) is difficult to maintain. Chip vendors face the problem of distributing models of new devices in such a way that designers can use these devices in systems before silicon is available, while at the same time protecting their intellectual property. All these problems have limited the use of simulation as an alternative to prototyping.

ALCHEMIST is a tool that addresses these problems. ALCHEMIST converts a graphical description of devices into VHDL or Verilog source code. These generated models are simulation models of the specified device, while the graphical description serves as the documentation of the model. The models are not synthesizable, thus protecting proprietary designs and intellectual property.

### ***ALCHEMIST: AN OVERVIEW***

ALCHEMIST uses familiar representations of device behavior as input: state diagrams, timing diagrams and truth tables. All input is graphical; there is no need to write VHDL/Verilog directly. The generated models are VHDL and Verilog source, and are directly simulatable with any simulator. The graphical input specification can be used as documentation, since ALCHEMIST produces PostScript® output. Flow-chart based test-benches can also be specified, easing the task of testing and verifying the models.

ALCHEMIST models can be full-functional or bus-interface. Bus-interface models (BIMs) can be defined as “bit-accurate and timing-accurate” models of a device’s interface. Bus-interface models will accurately model all of the external interface of the device, but may or may not model the internal functionality. For example, for a microprocessor model, a bus-interface model will correctly model all of the control signals and bus cycles (such as read and write cycles), but will not provide simulation of actual instructions. Thus, the actual values on its data and address buses (which can only be provided by the instruction being executed) will be arbitrary. The advantage of using BIMs is that they are considerably smaller and faster than a full-functional model, thus using much less resources during simulation. BIMs can be used to quickly verify that all devices work together in a system as specified. Even though the values on the address/data buses of a microprocessor are arbitrary, you can still place specific values on these buses, thus allowing you to test the system thoroughly.

Conceptually, ALCHEMIST divides a device into two parts, the bus-interface, and the internal circuitry that provides the functionality of the device. Figure 1 depicts this division. The ALCHEMIST BIM model simulates the bus-interface part of the device. This includes simulation of all the signals of the interface with correct timing checks and delays and simulation of all the transactions of the interface.

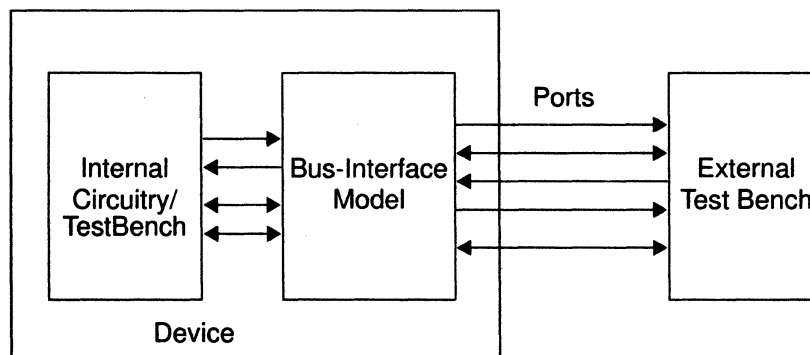


Figure 1 ALCHEMIST division of a device model

The device definition starts with defining the signals, which can be single or multi-bit. Timing information is represented by timing symbols, which are used throughout the model to represent timing constraints (setup and hold times, etc.) and output delays. Each device can have one or more part numbers, which are differentiated by different values for each timing parameters. Thus, part numbers are used to represent different speed grades of a device. Internal variables can be used to represent registers and memories in the device, which allows full functional modeling of a memory device.

The interface of a device is divided into one or more cycles. Each cycle represents a complete, independent transaction on the bus. For example, a PCI memory read transaction can be modeled as a cycle of the device. Cycles are associated with an initiation condition, which determines which cycle is simulated. Cycles are executed sequentially, and at the end of the current cycle, all the initiation conditions of the different cycles are tested to determine the next cycle.

Each cycle can have an optional state diagram, which indicates the states and transitions between them. Figure 2. is an example of a state diagram depicting a PCI bus master transaction. This shows the various states that the PCI master goes through while completing a transaction. The ADDRESS state is not a simple state, but a hierarchical representation for a group of states that manage the address phase of the PCI transaction. These include states that will terminate the transaction (for example, for a master abort, target abort etc.). The arcs connecting the states have boolean transition conditions associated with them.

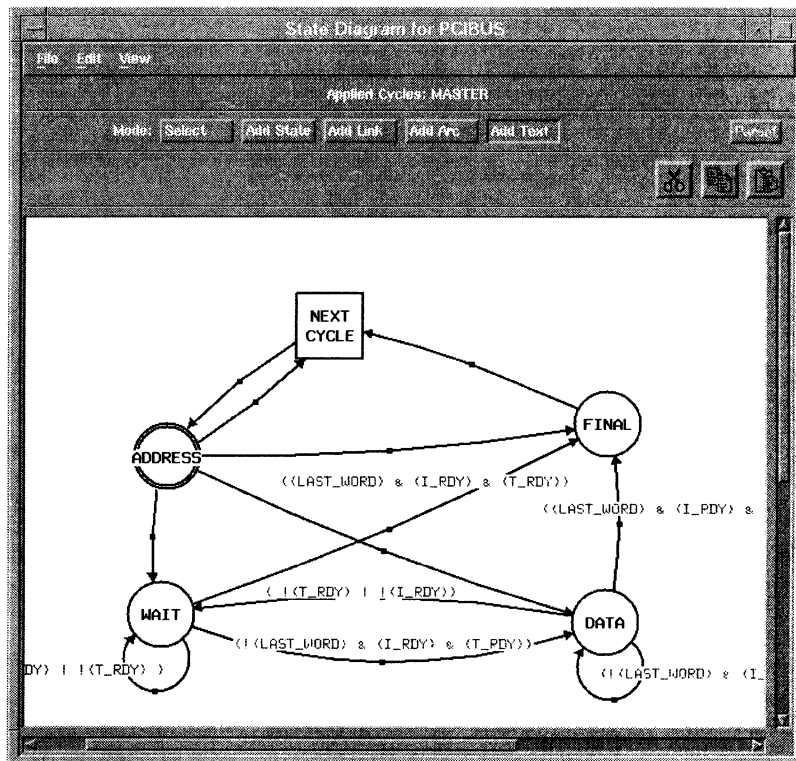


Figure 2 State Diagram of PCI Bus Master Cycle

The state diagram indicates the states and their transitions, but contains no timing information. The timing diagram for each cycle defines the timing and behavior of each signal. Figure 3 depicts the timing diagram for the master cycle of the PCI device corresponding to the state diagram in Figure 2.

The states shown at the top correspond to the states defined in the state diagram.

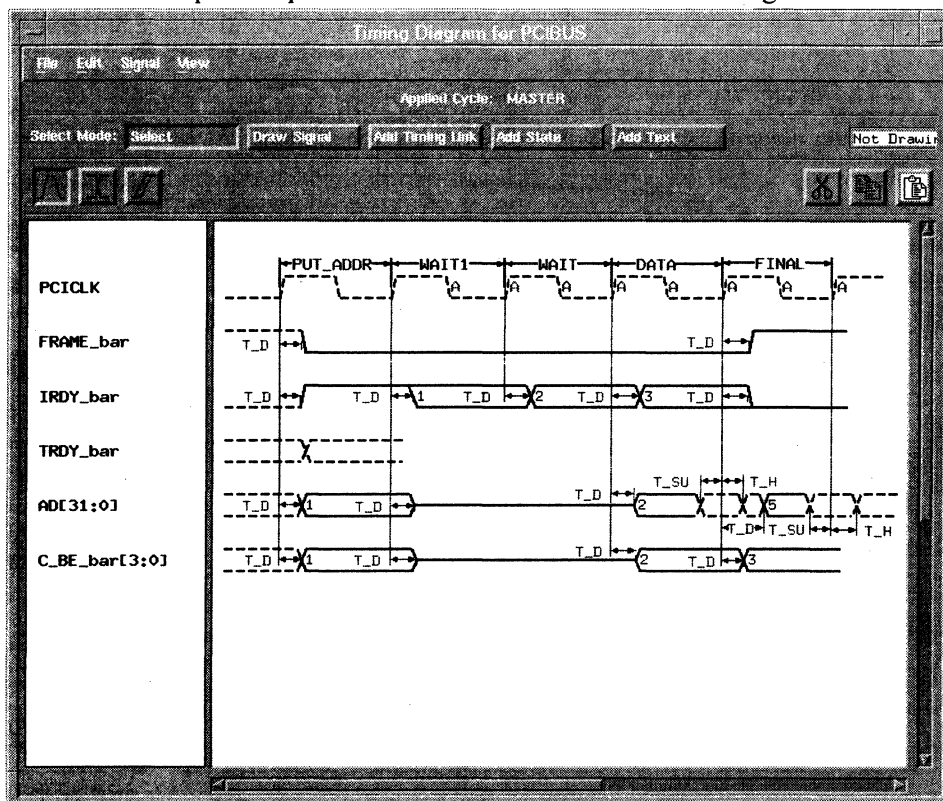


Figure 3 Timing Diagram of PCI Bus Master Cycle

The timing diagram indicates the different waveforms of each signal, and the timing relationships between the events on each signal. These timing links define setup and hold times, and output delays. The timing links are defined in terms of the timing symbols of the device. Values of output signals can be defined on the timing diagram, or they can be defined in truth tables.

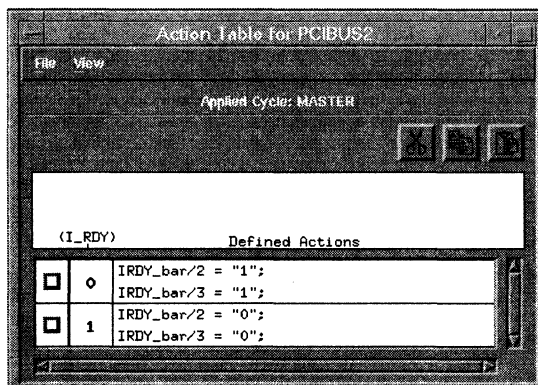


Figure 4 Action Table to determine value of IRDY\_bar signal

ALCHEMIST calls the truth tables action tables, because they define actions on output signals based on some combination of conditions. Action tables can have zero or more columns, which test for specific values on signals or a boolean condition. The different combinations of these column variables define the different rows of an action table. Each row can define a value for one or more output signals. Each timing diagram can have any number of action tables associated with it, to define various output signal values. Figure 4 shows a small action table that defines the output value of the IRDY\_bar signal,

determined from an internal condition depending upon the state of the device.

The above specifications can be used to define the bus interface of a device. However, these specification methods are not really sufficient to fully model a complex device, such as a microprocessor. For example, the instruction execution of a processor cannot be easily modeled using simple truth tables. But you can define a test-bench in ALCHEMIST to model such functionality. This test-bench (more properly called a model driver) takes the place of the internal circuitry shown in Figure 1. This test-bench is attached to the bus-interface model via control signals. The test-bench is used to sequence the model through different cycles, and to specify values to be placed on the various signals, such as address and data.

Figure 5 shows the flow-chart of a simple test-bench used to drive the PCI bus master device. This test-bench exercises various transactions on the PCI bus, providing the address and data for each transaction. The test-bench connects to the BIM through a simple interface. Each test-bench consists of one or more parallel processes, as well as any number of procedures. Defining procedures allows construction of a very modular test-bench, making it easier to construct complex test-benches.

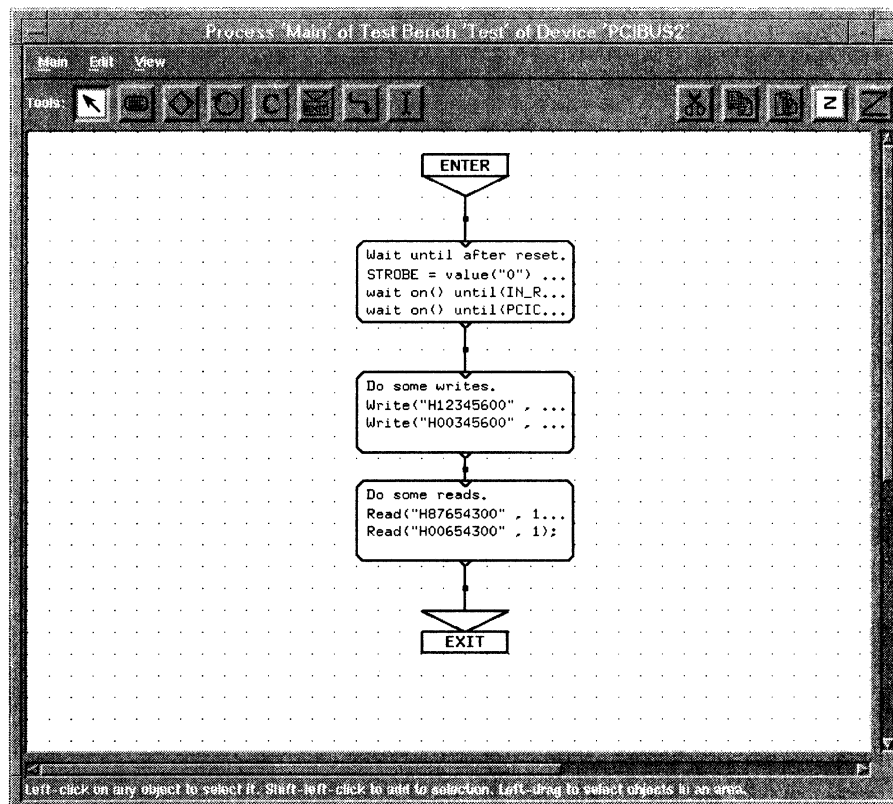


Figure 5 TestBench for PCI Bus Master Device

An external test-bench can also be defined for any device. This allows the user validate the model, and to exercise the bus interface of a device, particularly a device that is a bus slave, such as a memory or a PCI target. An external test-bench of such a device can be used to drive the various signals of the device, and observe the response of the device model.

Based on these inputs, ALCHEMIST generates VHDL/Verilog source code. The source code is ready for simulation. The generated BIM will correctly simulate all state and cycle transitions, check all input timing constraints (such as setup and hold times), and drive all output signals with the specified value and delay. In addition, the graphical input can be output as PostScript documentation, for use in documenting the device and model.

### ***ALCHEMIST MODEL USES***

ALCHEMIST models are not synthesizable. This means that IC vendors can distribute ALCHEMIST models of their new devices without risking intellectual property. These models can be created and distributed in advance of the actual silicon to provide designers with a way to create designs early, speeding up time-to-market. ASIC vendors can use ALCHEMIST to provide simulation models of their cell library, particularly for large cells such as embedded  $\mu$ C and DSP cores. The graphical input, such as state and timing diagrams, can be exported to word processors and the like for use in data sheets of the devices.

ASIC designers can create an ALCHEMIST model of their ASIC to be used as a reference document. The model can be used for simulation, before the synthesis of the ASIC is complete. This allows parallel development of the ASIC and the board. An ASIC test-bench can also be defined to exercise the device.

System designers can use ALCHEMIST to quickly create device models that are otherwise unavailable. These models will be created on an as-needed basis, reducing cost and delay. The models can be partial or complete, as necessary. The user has complete control over what goes into the model, thus making the models easy to change as necessary, as well as to debug.

ALCHEMIST models can also be used to design and verify bus protocols, for example the next generation of PCI.

### ***CONCLUSIONS***

Simulation is an increasingly important part of the design flow, but lack of models is a major hindrance to its widespread use. ALCHEMIST addresses this problem by providing a way of creating simulation models without VHDL/Verilog expertise. These models can be used in various ways to improve the simulation of boards and systems, to reduce the overall design cycle and time-to-market.

## VERIFYING PCI BUS SYSTEM AT MEGAHERTZ SPEED

Sanjay Sawant  
Quickturn Design Systems  
440 Clyde Ave.  
Mountain View, CA 94043  
(415) 694-6580/ (415) 967-3199 (FAX)  
e-mail: ssawant@qcktrn.com

*Abstract: The Peripheral Component Interface bus, perhaps the most defacto standard since IBM PC and Sun Workstations, is going into chips, boards and boxes. It is well accepted in many applications as makers of computer-related equipment hop on the bandwagon to share in PCI's well known price/performance gains. Designers charged with developing PCI products must make the right design choice and then right verification choice to get on to the bus. This presentation is focused on a verification methodology for PCI designs and slow down issues associated with it. It gives an overview of Logic Emulation and its application to PCI verification.*

### VERIFICATION CRISIS

Managing design complexity is becoming more challenging in the face of widely available submicron manufacturing capabilities, top-down design and the demand for the fastest time-to-market. As a result, the need for higher levels of design abstraction has become a key issue in submicron design. Eventhough, Hardware Description Languages (HDLs) have managed to generate millions of gates using logic synthesis, verifying gates using traditional software simulators has proven to be inefficient. Today's design methodologies have shifted towards concurrent hardware/software development. Majority of design teams are therefore staffed by hardware and software designers, both working in parallel on firmware design.

In addition to significant enhancement in the design complexity of PCI designs, designers are challenged by verification crisis. It is extremely difficult to thoroughly specify all system requirements to insure adequate test coverage as per PCI specifications. Traditional verification tools are too slow for complex designs. Many aspects of today's PCI designs can not be verified using event based simulators. This is because the performance of simulators change exponentially with the complexity of a design. For example PCI protocols and PCI applications that require real time video can not be verified using any software simulators. These applications require billions of clock cycles for system verification sign-off.

The verification problem is further compounded because designers not only have to verify their chips under development but also their interfaces with rest of the system. Above changes in the design methodology demands changes in the verification methodology.

### INTRODUCTION TO EMULATION

To bridge this verification gap a new technology has emerged called an Emulation. An emulation is a technology that enable designers to imitate logic of their design either in the form of a programmable devices or processors. As indicated in the *figure 1*, an emulator can be a resource shared on the network by multiple designers.



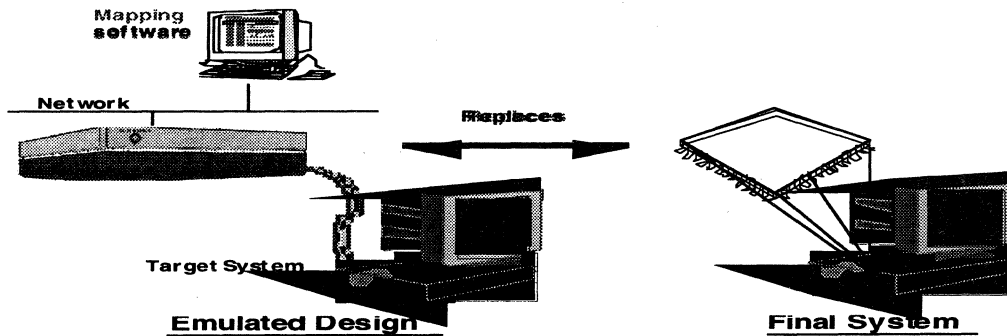
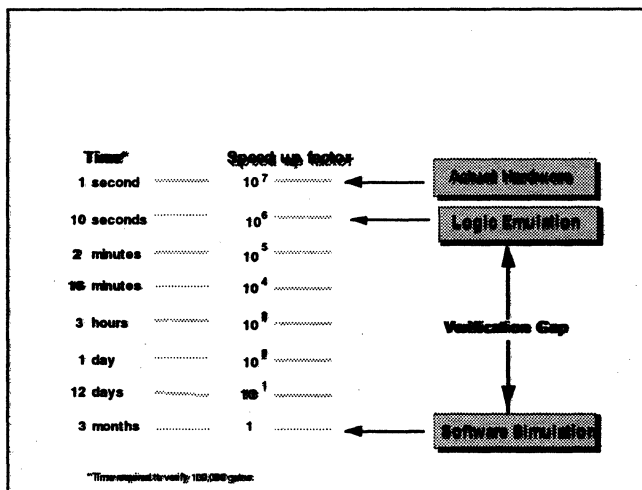


figure 1

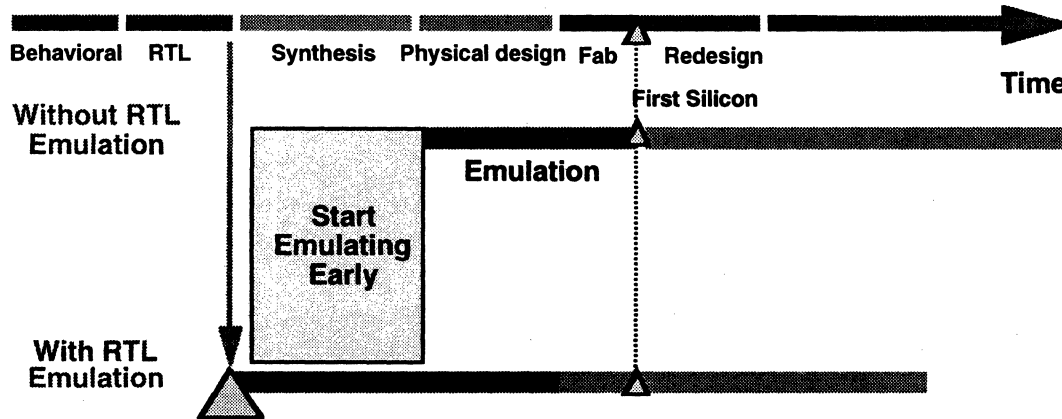
Emulation enable designers to automatically generate hardware prototypes of their designs. It takes away the burden of debugging the prototype so that designers can concentrate on debugging their design. This is a only technology that provides full system-level verification, up to 4 MHz speed, full design visibility and hardware/software co-verification capability. As indicated in the following speed chart, emulation runs orders of magnitude faster than traditional verification techniques.



## RTL EMULATION

Recently, a technology has emerged that couples fast RTL mapping with emulation. RTL emulation is directed at simplifying the emulation use model and to reduce time-to-emulation to bring emulation upstream where it can be used for performing architectural trade-offs. This enable PCI designers to validate their RTL in context of an entire system and verify whether RTL is really golden. This technology allows designers to visualize an effect of plugging their RTL into final target environment. Unlike gate-level emulation, designers need not wait for gate-level netlist to be available, or be concern about specific ASIC library, or Central Delay Calculators, or wire-load models. RTL emulation technology offers full visibility over a design through net name preservation and hundred percent readback. This enable designers to design and debug at RTL-level. Also quick design changes are made possible through incremental synthesis which is tightly coupled with incremental compilation. Most of the logic emulation systems have builtin logic

analyzer that allows designers to trace primary I/Os as well as internal nodes. Given below is an example of an emulation flow when RTL Emulation is used.



### SLOW-DOWN ISSUES

The beauty of PCI is its practical price/performance ratio. A typical PCI application runs at 33 MHz. One of the caviar in using logic emulation for PCI verification is building a target system and then slowing it down to few megahertz. The trick is to slow down the global PCI clock when it is used in a system running at a slower speed (approx. 1 MHz). Eventhough, the PCI Local Bus Specification does not recommend this frequency, it has been tested on several projects. Whenever PCI bus is used to transfer data from a master to an agent, the agent's clock is synchronized with the PCI bus clock. A typical case would be data transfer to the frame buffer of a graphics card. Also, the timing out is based on the number of clocks and not on the absolute time, which implies that, if you have an entire system running at low speeds, then only slowing down the PCI clock should solve the problem. Video cards are relatively easy to slow down. One can literally stop the bus without any ill effects. We have successfully slowed down a 720K floppy in a 1.44 Mbytes drive to approximately 500 KHZ. The slower read/write speeds keep the floppy from overflowing. Similarly, MFM and IDE controllers can be easily slowed down to 500 KHZ range. Finally, BIOS may need some modification as well. One may have to reprogram refresh timers in BIOS to increase bandwidth. An extra 150 KHZ margin can be gained by increasing the FIFO depth for floppy controller (to max 15).

Thus with an ability to verify PCI designs, in-circuit at megahertz speed, logic emulation has become next frontier in the system verification.

## MEASURING AND OPTIMIZING PERFORMANCE OF PCI BASED DESIGNS

Venkatesh Arunarthi  
Sand Microelectronics, Inc.  
1630 Oakland Road, A-103  
San Jose, CA 95131 USA  
(408) 441-7138/7538 (fax)  
e-mail: venkat@sandmicro.com

### **ABSTRACT**

There are design and verification tools available in the market which facilitate the development of PCI based designs. They include PCI synthesizable cores which accelerate development of PCI based ASICs and PCI bus simulation models which facilitate verification of PCI designs for compliance with the specification. However, no tools are available which enable designers to measure and more importantly optimize the performance of their PCI designs in simulation. In order to arrive at meaningful performance data, designers currently need to visually analyze the thousands of cycles of simulation data and manually calculate the performance numbers. This process can not only be very time consuming but also error prone. Therefore, designers tend to randomly spot check the simulation data to get a general idea of the performance. However, this method is highly inaccurate since the performance calculation is based on a few cycles of simulation data and not on the entire simulation run. Furthermore, design changes which impact performance could be overlooked, resulting in silicon being fabricated that may be functionally correct but may not meet the performance requirements. This paper presents a method for analyzing and optimizing performance of PCI designs in simulation prior to silicon fabrication.

### **TRADITIONAL DESIGN FLOW**

In the 1980s, most ASIC-based systems designs used a low-level design methodology: ASIC foundries provided customers with a set of primitive functions represented by gates. These gates became the building blocks used to develop digital systems. By interconnecting these gates, designers were able to develop application-specific functions. This approach is rapidly becoming obsolete as designs are becoming more complex.

This low-level design methodology has two primary disadvantages:

1. The design is tied to a technology or foundry, since the designers are using foundry-specific building blocks. Therefore, changing the technology or foundry becomes difficult and time consuming.
2. The more complex the design, the more difficult it is to understand. It is difficult for designers to keep track of a complex design and it is even more difficult for someone other than original designer to understand the design, since there may be a large number of gates interconnected with each other.

ASIC-based system design methodology has changed for the better, as we enter the next millennium. Designers, today, are using a high-level design (HLD) methodology and hardware description languages (HDLs) such as Verilog and VHDL to describe a design's functional behavior. The design is simulated and checked for functionality using the simulators provided by HDL vendors. Designers typically write numerous test cases for verification, and use the waveform display tools for debugging. Next, the design is synthesized and mapped into one of the foundry specific libraries. Finally, the gate level netlist is simulated to make sure that the design meets the timing requirements before tapeout. This approach has several advantages:

1. Engineers can be more productive if they design at higher levels of abstraction. They can add value or analyze tradeoffs quicker at behavioral level than at the gate level.
2. By using a high-level design approach, designers can express design functionality in a technology independent language. And, instead of having the design based on a single foundry or technology, designers can choose from multiple foundries and technologies.
3. Hardware description language (HDL) usage ensures an easier to understand and better documented design.

### ***THE NEED FOR AUTOMATIC PERFORMANCE ANALYSIS***

Changes to system design methodologies have resulted in the introduction and continued improvement of tools for designers. For example, HDL simulators, synthesis tools, and timing analysis tools help analyze a design's functionality and timing, but they do not measure performance of the design before silicon production.

Designers have been left to their own resources to analyze and predict performance. They visually analyze simulation data and waveforms cycle by cycle, and manually calculate performance numbers. Unfortunately, simulation data can represent anywhere from a few hundred cycles to a few million cycles of simulation. Done manually, this is time-consuming and error-prone. Therefore, designers randomly spot check the simulation data to get some idea of performance.

This approach is not accurate because performance calculations are based on only selected cycles of simulation data, and designers could miss simulation cycles that significantly impact performance. Also, during the design process, several design changes are typically made including ones that can impact performance. Very rarely does a designer think to go back and re-calculate performance.

Once actual silicon is available, there are commercial tools that evaluate performance. The main drawback to having performance data at this point is that the system's performance may not be what was expected and it is too late to change the design. The designer options are accepting lower performance or making changes to improve performance by going through another design iteration. This typically takes months, can cost \$25,000 or more for another round of silicon fabrication, and delays a product's introduction.

### ***PCI AND THE PCI MARKET***

PCI is the solution for performance-hungry applications. PCI-based systems usually include graphics, video, disk and network cards. PCI design starts are increasing dramatically. Published Dataquest figures indicate that from less than a million units of PCI silicon in 1993, the market is expected to expand to more than 49 million units in 1996.

Developing a PCI-based component is a complex exercise. The specifications are very strict. Therefore, the growing PCI market is compelling EDA suppliers to provide system designers with easy-to-use PCI bus models. These pre-built, pre-tested models have relieved designers from the "will it work?" pressures. "Is this best possible design?" then becomes the next question.

### ***THE PCI DESIGN PROBLEM***

The PCI bandwidth of 132 Mbytes/sec. is sufficient for most applications, including real-time video. However, designers must ask if enough bandwidth is available to manage multiple simultaneous

applications in a system-level configuration. For example, system latencies such as arbitration and wait states due to variations in fifo-depths affect available bandwidth.

**WHAT IS PCI PERFORMANCE ANALYZER?**

The proposed Sand's patent pending PCI Performance Analyzer (PPA) allows the designer to evaluate the performance impact of changes in the parameters of their design, such as the number of masters, FIFO depths, arbitration schemes, latencies and wait states. The designer can evaluate the impact on the PCI bandwidth of adding additional components.

**PPA ARCHITECTURE**

In a typical PCI design simulation environment the designer uses a PCI Bus Functional Model to generate bus cycles against his/her Design Under Test. The PPA connects to the PCI bus as shown below. It consists of three major blocks, a Monitor, a Parser and a Graphical User Interface.

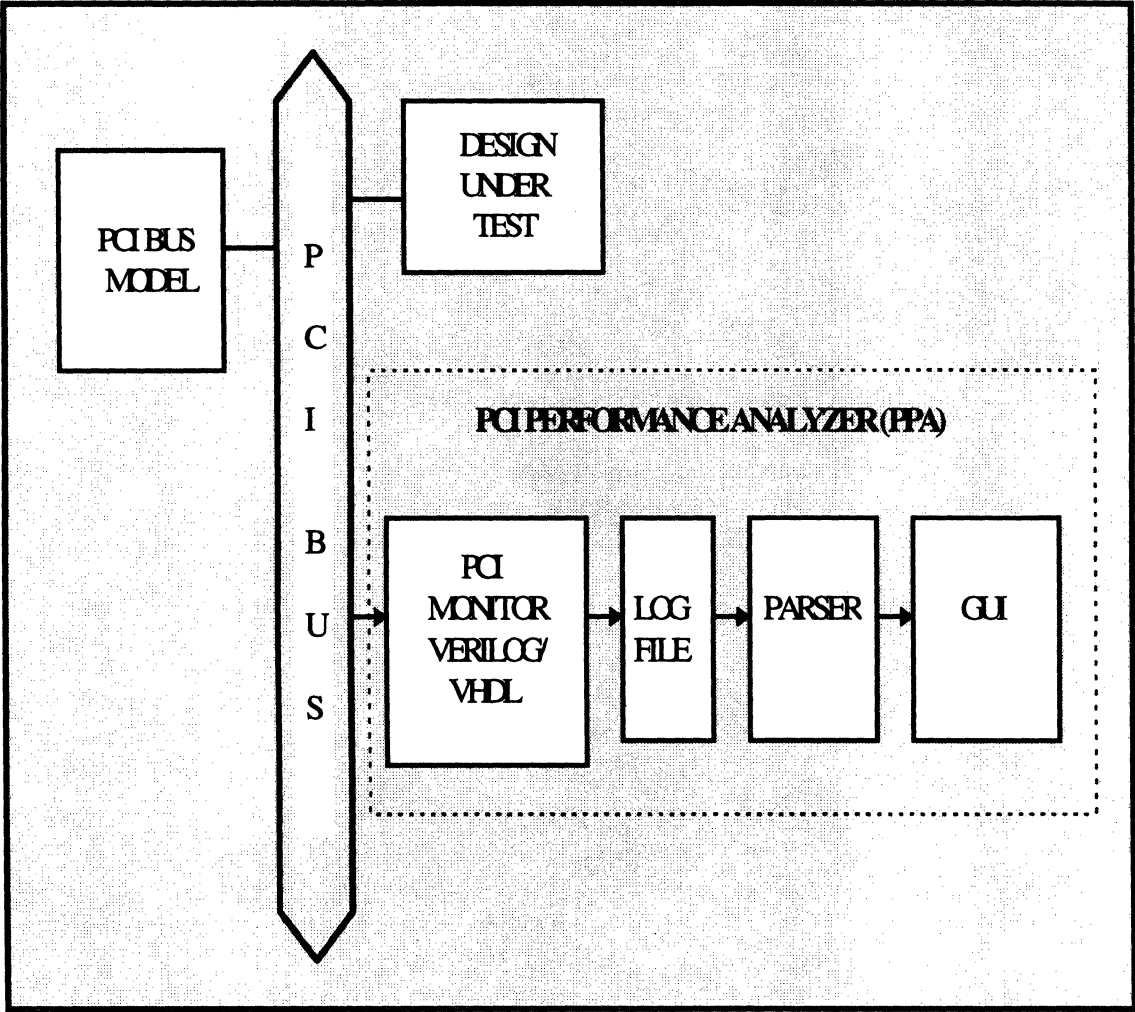


Figure 1 Typical Simulation Environment

## Monitor

The Monitor was implemented in Verilog and VHDL to support both the simulation environments. Monitor connects to the PCI Bus and logs the states of all the PCI signals on every rising edge of the clock into a file. This file is then analyzed by the Parser to calculate the performance data. In addition to logging, the Monitor also checks for PCI Rev2.1 Compliance and reports all the Protocol Violations and Timing Violations into two different files, which will later be displayed through GUI.

## Parser

The Parser was implemented in C. It parses the log file created by the Monitor, one transaction(burst/non-burst) at a time, and stores the parsed data in a Linked List Data Structure. This linked list is then analyzed to extract the relevant information needed for the subsequent calculations. As each transaction is extracted, a set of global data structures are updated with the transaction data like Command, Address, Wait States, Number of bytes, Transaction begin and end times etc. This procedure is repeated, until the entire log file is parsed. Finally data is filtered according to the user configuration provided through GUI, and stored in yet another set of data structures, ready to be display by GUI. User configuration will be explained in the following section.

## Graphical User Interface (GUI)

The GUI is an user friendly front-end tool, developed using Motif. The GUI supports both SUN & HP platforms. Depending on the type of the data it provides, the GUI can be divided into three different windows, namely Statistics, Performance, and Protocol & Timing windows. Each of these will be explained below.

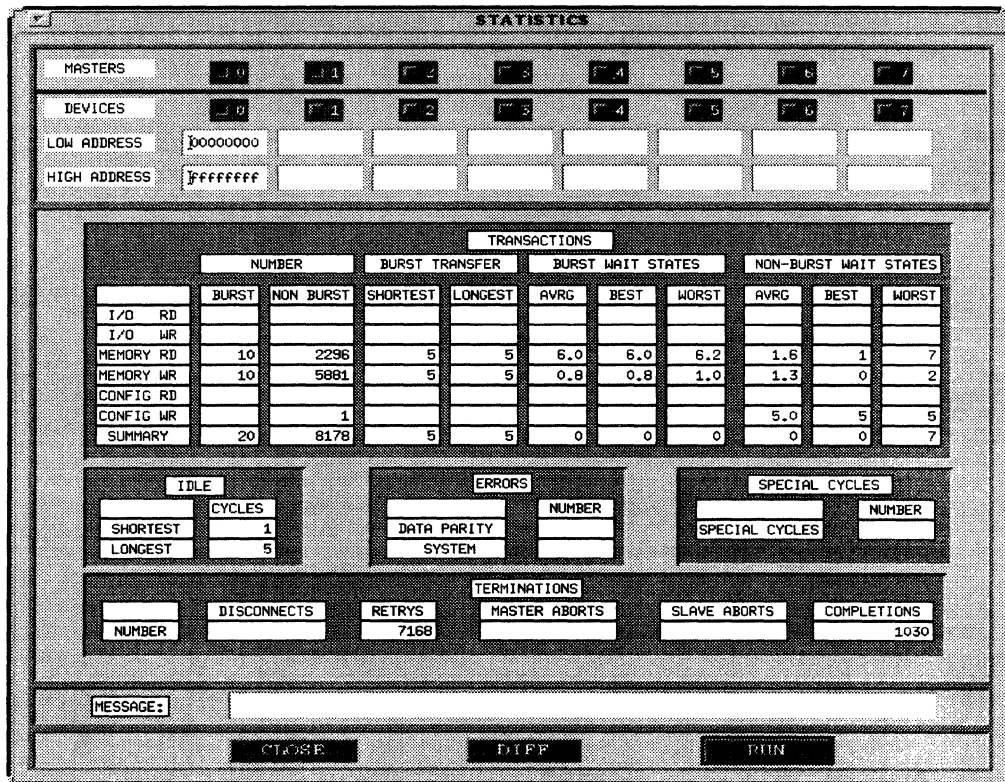
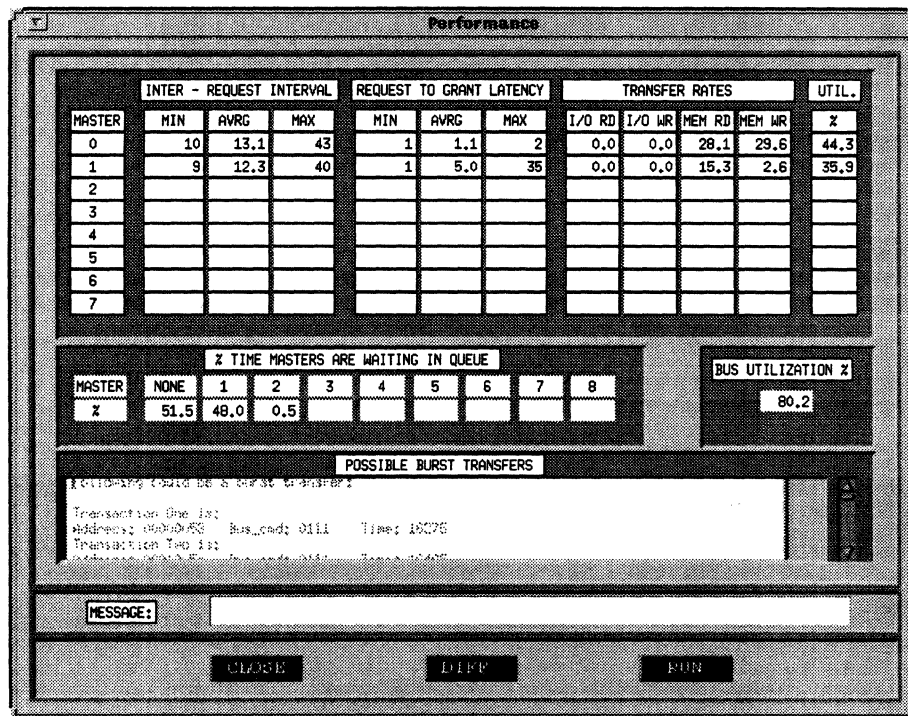


Figure 2 Statistics Window

**Statistics Window:** The Statistics Window provides the designer with an exhaustive statistical analysis of all the possible cycles on the PCI bus for a particular simulation run. Generated data includes information such as best and worst case wait states, number of read and write cycles, and abnormal termination conditions. This window features a user configurable sub-window, where designer has the options to display all the cycles that took place on the bus, or selectively look at the data on a target by target basis/master by master basis. This enables the designer to look at the data related only to his/her design in a system level simulation environment, where multiple blocks from different designers are combined. Instead of watching data for entire simulation, designer can choose to display information that belongs to a particular time-window in simulation. The statistics window also provides event time stamps that provide links back to the simulation environment to help debug the design.



**Figure 3 Performance Window**

**Performance Window:** This window mainly focuses on the performance data related to arbitration schemes and bus band width. It displays the data, such as how frequently PCI bus masters are getting on and off the bus, and what data transfer rates they are achieving. It also provides other information such as request to grant latency, individual bus utilization of all the bus masters, and percentage of total simulation time masters are waiting in queue to get onto the bus.

**Protocol & Timing Window:** If during simulation, the PPA detects a Master/Device violating the PCI Protocol or PCI Timing, all such errors are flagged for review in these windows.

## ***CONCLUSION***

A method has been presented to measure and optimize the performance of PCI based designs in simulation environment, prior to the fabrication. Though current implementation focuses only on the PCI bus, this concept can be readily extended to other I/O buses, process buses, and memory buses.

## ***BIOGRAPHY***

Mr. Arunarthi is currently working as a design engineer at Sand Microelectronics, Inc. He holds an MSEE from University of Southwestern Louisiana. He was responsible for the development of the PCI Performance Analyzer at Sand Microelectronics. His other accomplishments include development of simulation models and synthesizable core products for PCI Bus and Universal Serial Bus.



# A VHDL Design Approach to a Master/Target PCI Interface

Leo K. Wong  
Applications Engineer  
Altera Corporation

Martin Won  
Applications Supervisor  
Altera Corporation

Subbu Ganesan  
Associate Director of Hardware Engineering  
ZeitNet, Inc.

## ABSTRACT

*This paper describes a design approach to implementing a Peripheral Component Interconnect (PCI) interface that allows for the maximum amount of design flexibility while achieving an actual working solution in a relatively short amount of time. The approach involves two key elements: VHDL and programmable logic devices. The portability of VHDL and the rapid prototyping time of programmable logic, combined with the flexibility afforded by both creates a design approach that provides the designer the opportunity to make changes to the design while still working towards a final hardware solution. In the experience of the ZeitNet project (an interface for an ATM adapter card), this approach yielded a demonstratable product in four months; in another three months, burst mode was added to the design and final testing was completed, resulting in a finished product in only seven months from product inception. Furthermore, considerations for future development of PCI interface are also included.*

## 1. INTRODUCTION

Most engineers are faced with the challenges of ever shorter production cycle, higher performance requirements as well as cost pressure in every project. A well defined design methodology is critical in meeting these goals. Our sample design is a PCI bus ATM adapter card. Table 1 shows the requirements of the project.

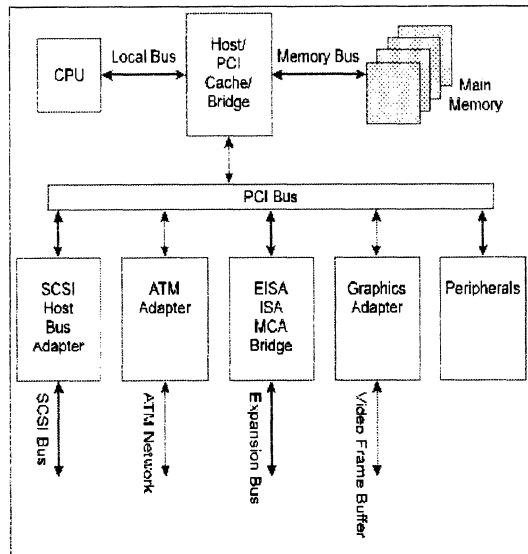
**Table 1 Project objectives**

High Performance	Full PCI and ATM compliance; Zero-wait-state Burst transaction; Sustaining full duplex line speed
Interoperability	Product should be accepted by multiple platform for maximum customer appeal.
Vendor Independence	Need flexibility to migrate to future silicon technology if desired.
Meet product rollout deadline	Three to four month design cycle time limit from concept to silicon.

The following sections will discuss these objectives in details and explain how these objectives are met by the proposed methodology.

## 2. ARCHITECTURE CONSIDERATIONS

Figure 1. Typical PCI Bus System with ATM Adapter



### 2.1 Performance

Bus bandwidth is important not only to networking performance but also system performance. The PCI Bus is capable of high performance data transfer through its high bus bandwidth capacity. The maximum PCI Bus transfer rate is:

Clock Frequency	= 33Mhz
Bus Width	= 4 Bytes = 32 bits
Max Transfer Rate	= 133MB/sec
	= 1.06Gbit/sec

The SONET 155 Mbps ATM requires 134 MBps transfer rate, significantly less than the maximum PCI bus transfer rate.

Performance, however, does not depend on bandwidth alone. In order to realize the full potential of the PCI bus, burst transaction is expected to be implemented by the interface. PCI Bus specification enables variable burst transaction size. The interface component should be able to handle variable burst size.

Moreover, a low bus latency is necessary to provide a quick turnaround time. The overall bus latency is comprised of three parts:

- Arbitration latency - the time the Master waits after asserting REQ# until it receives GNT#
- Bus acquisition latency - the amount of time the device waits for the bus to become free after GNT# has been asserted.
- Target latency - the amount of time that the target takes to assert TRDY# for the first data transfer.

While the ATM adapter project is a Master and Target combined PCI interface, all three types of latency should be taken in careful consideration. The PCI interface component is challenged to implement the design that meet the aforementioned performance requirements.

### 2.2 Interoperability

To ensure the widest possible market acceptance, products should be accepted by as many platforms as possible. As a bus architecture, PCI promises processor independence. However, due to the evolving nature of the PCI architecture, there are systems that does not adhere strictly to the latest standard. It is highly desirable to have a versatile PCI interface component to implement the required modifications in accordance with the operating platform.

### 2.3 Vendor Independence

Depending on the market demand and thus production volume, engineers should have the flexibility to switch from one silicon technology to another. For instance, at mid-to-lower volume production, programmable logic device is ideal for its flexibility, zero NRE cost and low inventory risk. In high volume production, it might be more cost effective

to migrate to a Masked Programmable Logic Device (MPLD) or an ASIC solution.

An ideal engineering methodology should provide a quick migration path to the most cost effective silicon solution in reaction to market demand. Proven transition path from one silicon technology to another should be provided.

## **2.4 Design cycle**

The ATM adapter project was under tremendous time pressure. The month was March, and ZeitNet was scheduled to demonstrate their ATM adapter card at the Tokyo Interop show in July. There were fourteen weeks available from product definition to silicon realization.

## **3. System Methodology**

To achieve the project's challenging goals: fully PCI and ATM compliant, low cost and flexibility within three-to-four months, designers must weigh several inter-dependent aspects of their engineering cycles. Critical to a project's success are the design entry method, EDA tools and the silicon choice.

### **3.1 Hardware Selection**

At the time, to implement a PCI interface for the ATM card, there are mainly two selection: PCI chipsets or programmable logic device.

Off-the-shelf PCI interface ASIC or PCI chipsets decrease the resources required for in-house development, but the ones available on the market lacked the flexibility for customization. Due to this shortcoming, the chipsets were deemed inappropriate for the project.

### **3.2 Design Entry**

An industry standard high-level hardware description language is desirable to ensure smooth future migration in technology. VHDL satisfied the need because of its wide acceptance in the EDA community.

While designers usually need to instantiate device specific primitive for optimal performance and area results, careful modularization can lead to high degree of design re-use in future silicon technology.

By modularizing design, designers separate the universal behavioral code from the device specific primitives instantiation. The behavioral core, written in VHDL, can be re-used in other synthesis tools when porting to other silicon technologies. While the primitive instantiations maintain close and effective control over the interface component.

### **3.3 PLD Selection**

The next decision was to choose a programmable logic device that could implement a combined master/target PCI interface within a reasonable amount of time. Among the range of PCI-compliant devices offered by programmable logic vendors, FLASHlogic devices and MAX 7000E devices from Altera, and XC7000 EPLDs from Xilinx were explored.

The first concern for a programmable logic implementation was fitting the entire combined master/target interface into a single device. The FlashLogic devices were examined and their logic capacity is deemed insufficient to fit all functionalities into the largest member of that family, although it did offer several features that were attractive for PCI interface design, including very predictable timing, on-board RAM, and open-drain outputs. The same resource limitations seemed to hold true for the XC7000 devices from Xilinx, in addition to suspicions that

the critical timing required for the PCI interface would be difficult to achieve in those devices.

The final potential set of devices proved to be the ideal choice: MAX 7000E. By estimation, the largest devices from the family would accommodate a combined master/target design.

### 3.3 EDA tool selection

Traditionally, there has always been tradeoff between design abstraction and efficient silicon control. On one hand, using a proprietary semiconductor vendor tool provide efficient design and synthesis support for the specific component, but it is usually difficult to port the design to other technologies. On the other hand, by choosing a standard EDA design platform, designers risk sacrificing the tight integration, but gain the ease of migration to various ASIC or gate array technologies.

The development tools chosen for this ATM project is MAX+PLUS II, which includes a VHDL compiler. The tool can directly accept VHDL text entry, synthesize, place & route, simulate and generating programming file for MAX 7000E device without the burden of third-party tool translation. This design flow provides tight integration, allowing quick design changes and iterations. Moreover, MAX+PLUS II offers proven migration path interfacing with major third party EDA tools through EDIF netlists and vendor libraries.

To simulate the design on a board level, tools from Model Technology was employed. The process of developing the VHDL code required for the design took about 2 weeks; simulation was completed one month later.

## 4. IMPLEMENTATION

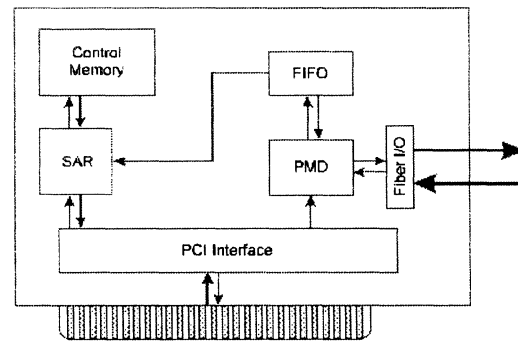


Figure 2. Zietnet PCI Bus ATM Adapter

### 4.1 Implementing Functionality

In order to shorten the design cycle, VHDL design code was developed in parallel with the device selection process; the goal was to work towards creating a functionally correct VHDL design using a VHDL simulator, and by the time of its completion, place the design into a device.

After a month of simulating the VHDL design and fine-tuning its functionality, ZeitNet was ready to fit the design into their chosen device family. The date was nearing the end of May, which left the month of June and some of July to fit the design, lay out the PCB, and test the overall product. In order to reduce the development time, it was decided to proceed with the design without implementing the burst mode, since it was not deemed absolutely necessary to demonstrate the basic functionality of the product in July. After the show in July, ZeitNet's engineers would revisit the design and add the burst mode. There was, of course, an amount of risk associated with this decision, but ZeitNet's engineers remained confident that they could add the burst functionality to the MAX 7000E device without negatively impacting the overall product.

Compiling their VHDL with MAX+PLUS II revealed that the MAX 7000E device required would be the 256-macrocell EPM7256E in the 208-pin QFP package. Without the burst-mode

ality, the design occupied about 75% of the device's logic resources and used about 100 pins.

### **Implementing Burst Mode**

After exhibiting their product at the Interop show in July, ZeitNet set about adding burst mode to their PCI interface. During the initial testing of their card, they discovered that most existing systems used host bridges that limited transfers to host memory. Specifically, the limits were: 32 bytes for a burst read cycle and 16 bytes for a burst write cycle. They designed their burst mode for 32-byte capability.

This segment of the design process took a little over a month and a half, with much of the time devoted to ensuring that burst capability would function in all tested platforms. These platforms include different PCI machines such as Compaq, Dell, DEC-PC, Gateway, Micron, NEC PC and various other clones. With burst mode, the entire combined Master/Target interface design occupied 220 macrocells, or about 86% of the EPM7256E device. Even with the increased utilization, the ZeitNet designers were able to keep the same pinout for the EPM7256E, and eventually brought the completed ZATM PCI-bus ATM adapter card to market at the end of October.

## **5. PCI EXPERIENCE**

The proposed platform: VHDL design entry and programmable logic silicon implementation successfully meet all goals set forth at the beginning of the project.

On the side of PCI Bus, the ATM adapter card was able to implement variable burst size transaction. In addition, zero wait state read and write transaction was also achieved, providing the lowest possible bus latency. These abilities actualize the full performance potential of the PCI bus. On

the backend ATM network, full duplex line speed was sustained.

In terms of migration ease, with the help of the tool, designers estimate that they would be able to re-use 80-90% of their VHDL code to port their design to an ASIC in the future. The versatility of the proposed platform was proven as the same design was re-targeted for another programmable logic device, an EPF8820A, a member of the FLEX 8000 PLD family, later in the production cycle.<sup>1</sup>

More importantly, Zeitnet was able to achieve all goals within the specified time frame: the product met the trade show demonstration as well as the production deadline.

## **6. FUTURE ROADMAP**

Looking forward, there are several paths of modifications, mostly related to the evolving nature of the PCI standard and systems offering PCI compatibility.

- (1) As noted earlier, the host bridges in most of the tested systems limited transfers to the host memory. The adapter card was designed accordingly, but future host bridges are expected to provide larger transfers in the future.
- (2) None of the tested systems had implemented the latency timer. Correspondingly, no latency timer was implemented in the adapter card. This functionality is expected to be added when latency is supported by more systems.

These are functionality concerns for the future of PCI as an evolving architecture. Meanwhile, the interface component should be versatile enough to handle these modifications.

---

<sup>1</sup> While MAX 7000E is a AND-OR gates-based CPLD built on E<sup>2</sup>PROM technology, FLEX 8000 is a Look-Up Table based SRAM technology.

## **Conclusion**

From this case study, one can conclude that all engineering challenges were met. While the final product not only meet release schedule, it also attain high performance and maintain a versatile future growth path.

It is obvious that the benefits of the proposed methodology can readily be extended to other areas of electronic engineering. Flexible engineering control such as shortened time-to-market, versatile volume adjustment, and vendor independence; high performance silicon technologies coupled with easy to use software are universal advantages all designers should utilize.

## **Author biographies**

Leo Wong is an applications engineer with Altera Corporation (San Jose, CA), where he is involved in PCI design implementation, megafunction partners liaison, and programmable logic applications consultation. Leo holds degree in Electrical Engineering and Computer Science from University of California at Berkeley.

Martin Won is the Applications supervisor of Technical Communications for Altera Corporation, where he has worked for 5 years. His responsibilities include writing and publishing articles, creating and presenting Altera's technical seminars, and producing Altera's quarterly newsletter for its customers. Mr. Won holds a B.S. in Electrical and Computer Engineering from the University of California at Santa Barbara.



High Performance System Controllers and Data Communications Controllers

# **The GALNET Architecture: A PCI-Based Solution For High Performance Internetworking**

Galileo Technology, Inc.  
1735 N. First St. #308  
San Jose, CA 95112  
Tel. 408-451-1400  
Fax 408-451-1404

---

APRIL 1996

1

## **Contents**

- The GALNET Architecture
- The GT-48001 Switched Ethernet Controller
- Architectural Overview



---

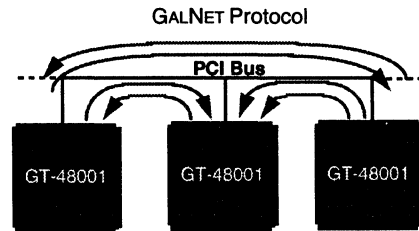
APRIL 1996

2

## The GALNET™ Architecture



- An architecture that allows for easy interconnection and expansion of networking chips over the PCI bus
- A family of GALNET chips is being developed by Galileo
  - The first is the GT-48001 Switched Ethernet Controller (SEC)
    - ▲ 10/20Mbps, 8 ports
  - The second will support 100Mbps
    - ▲ Introduction in 2Q96
  - Several others will follow
    - ▲ Various LAN and WAN technologies
- Simple protocol used in the SEC
  - GALNET protocol
  - 5 instructions



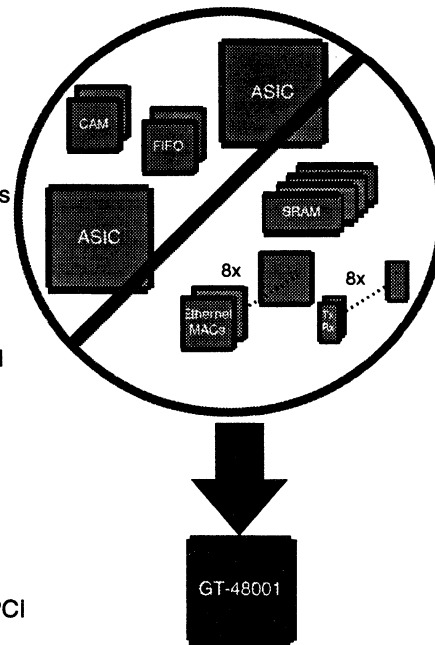
APRIL 1996

3

## GT-48001 Key Benefits



- High integration
  - 300,000 gates, 1.5 Million transistors
  - A complete system on a chip
  - 8 10/20Mbps ports in a 208PQFP
- Lowest cost in the industry
  - Integrated serial transceiver logic requires inexpensive external electronics per port
  - Uses standard DRAM instead of SRAMs or CAMs
- High performance
  - Filtering and forwarding at full wire speed
  - Minimizes dropped packets
- Intelligent switching and management
  - Enhanced network management
  - Differentiation via management hooks
- Easy expansion and connectivity
  - Expand to 256 ports
  - Interface to other LAN technologies via PCI

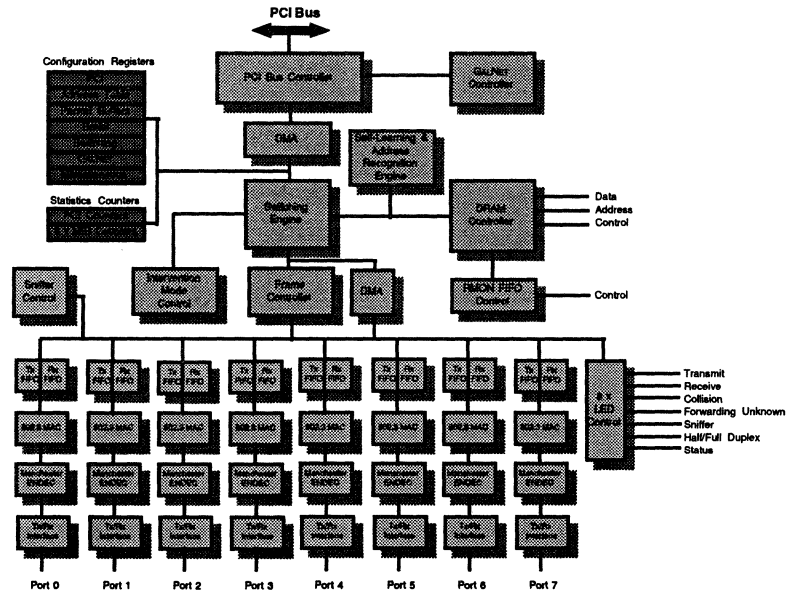


APRIL 1996

4



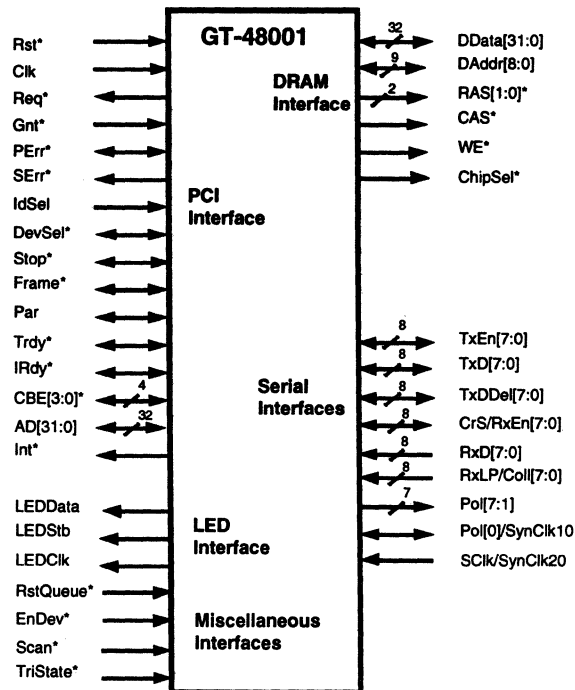
# GT-48001 Switched Ethernet Controller



APRIL 1996

5

## Pinout



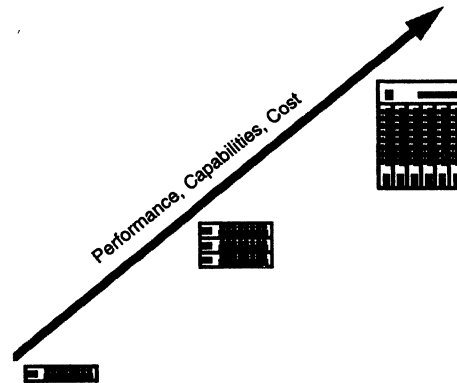
APRIL 1996

6

## Wide Product Spectrum Feasible



- The GT-48001 is a flexible building block
  - ➔ Departmental to enterprise
- Various system capabilities possible
  - ➔ Low cost unmanaged switches
  - ➔ Stackable switches
  - ➔ Low cost managed switches
  - ➔ High performance systems with sophisticated management
- High product differentiation possible
  - ➔ Many management hooks provided
  - ➔ More engineering applied --> more sophisticated product produced



APRIL 1996

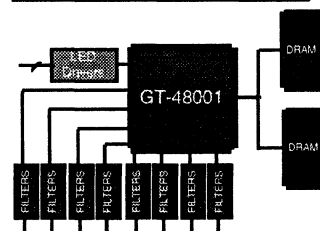
7

## Unmanaged Switches

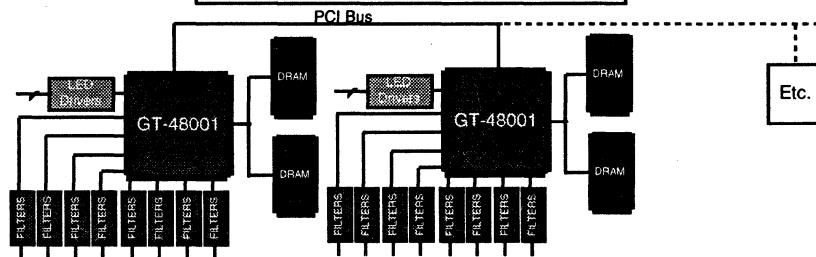


- Simple implementation
  - ➔ Add DRAM
  - ➔ Add serial interfaces
  - ➔ Add LEDs
  - ➔ Add power supply and case
    - ▲ GO!
- Lowest cost in the industry
- Simple expansion via PCI bus
- Auto-learning mechanism

8-Port Unmanaged Switch



16-Port to 48-Port Unmanaged Switch



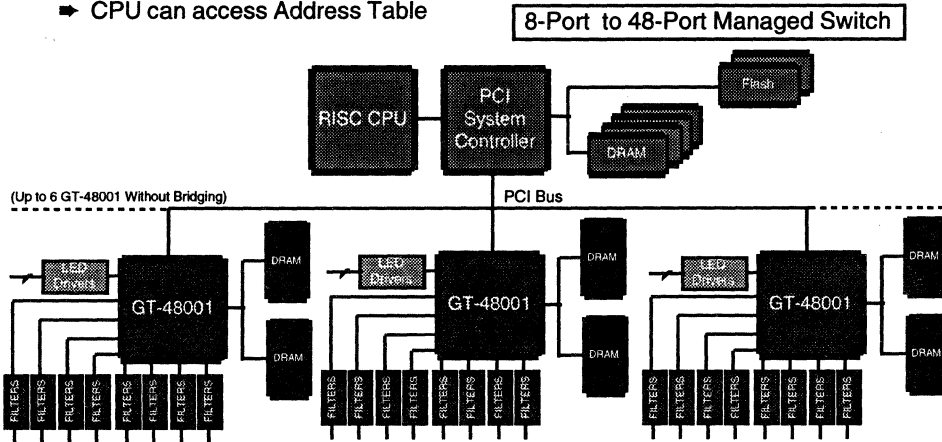
APRIL 1996

8

## Switch Expansion (< 48 Ports)



- Simple expansion by connecting devices via the PCI bus
  - Full wire speed, full duplex switching, guaranteed for up to 6 GT-48001's
- Add management via PCI
  - CPU can access Address Table



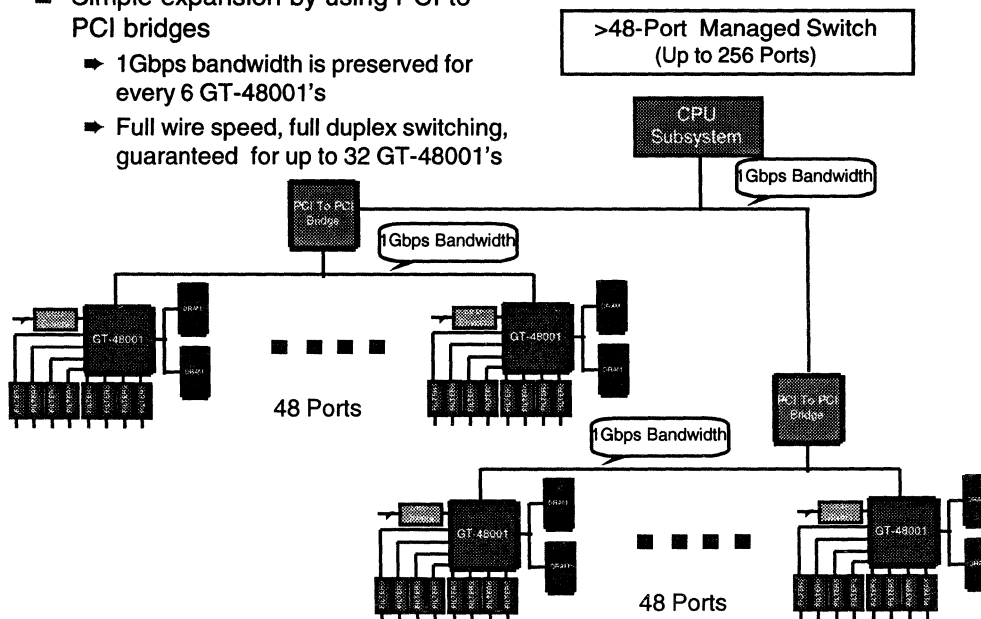
APRIL 1996

9

## Switch Expansion (> 48 Ports)



- Simple expansion by using PCI-to-PCI bridges
  - 1Gbps bandwidth is preserved for every 6 GT-48001's
  - Full wire speed, full duplex switching, guaranteed for up to 32 GT-48001's



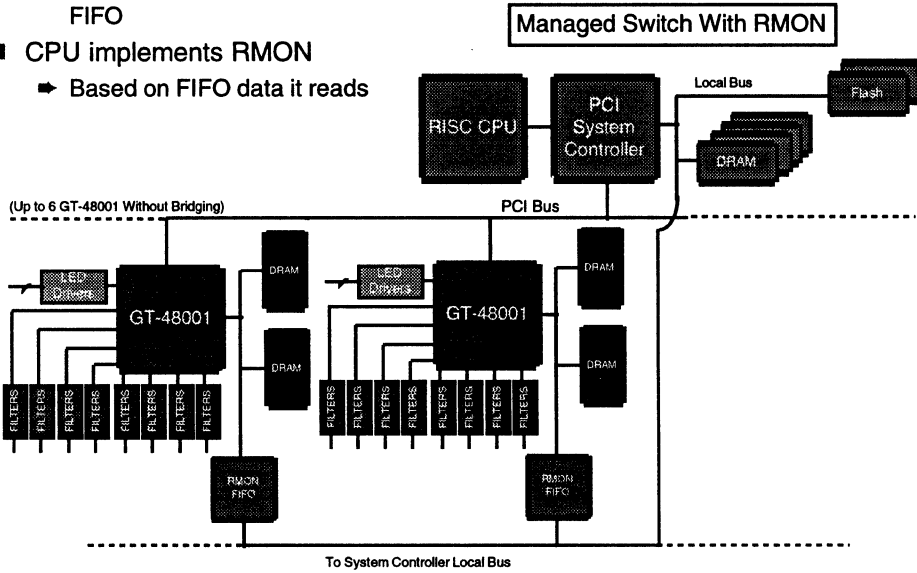
APRIL 1996

10

## Switch With RMON



- Control storage of connectivity matrix
  - The GT-48001 controls an external FIFO
- CPU implements RMON
  - Based on FIFO data it reads



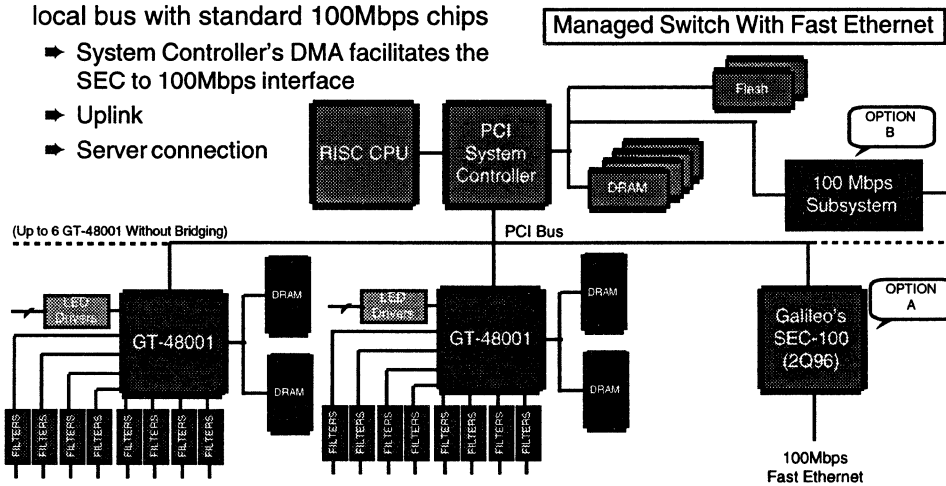
APRIL 1996

11

## Connectivity to Fast Ethernet



- Connect with GALNET family member SEC-100
  - Sampling 2Q96
- Alternatively, use high speed CPU local bus with standard 100Mbps chips
  - System Controller's DMA facilitates the SEC to 100Mbps interface
  - Uplink
  - Server connection



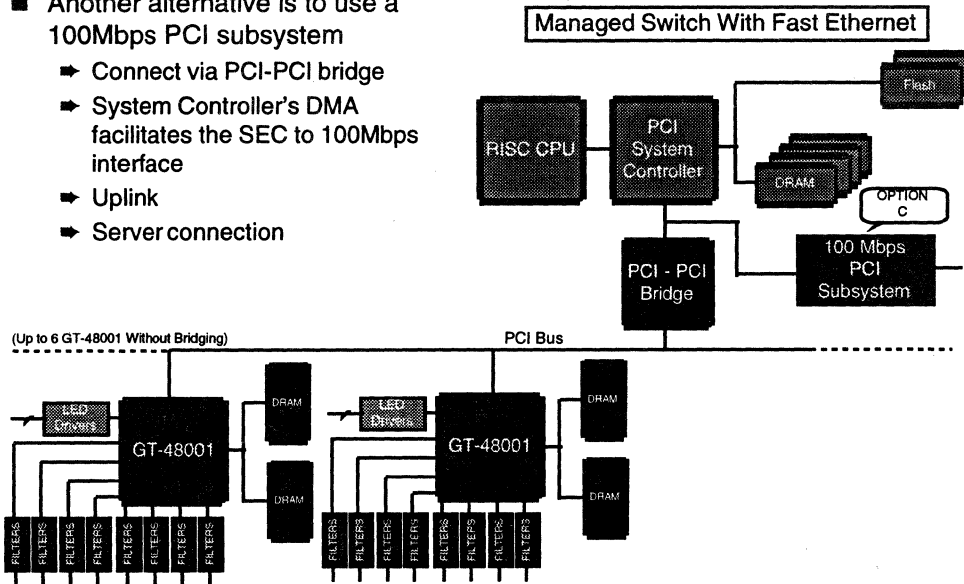
APRIL 1996

12

## Connectivity to Fast Ethernet (cont'd)



- Another alternative is to use a 100Mbps PCI subsystem
  - Connect via PCI-PCI bridge
  - System Controller's DMA facilitates the SEC to 100Mbps interface
  - Uplink
  - Server connection



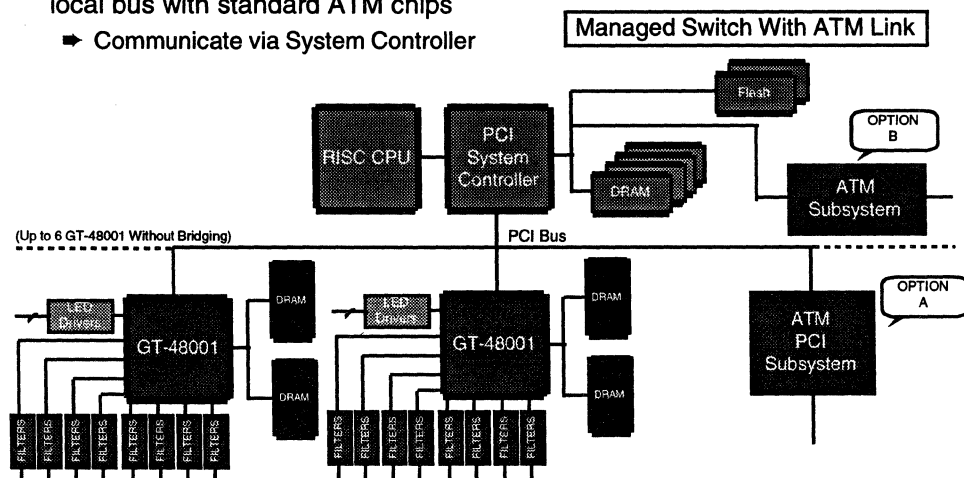
APRIL 1996

13

## Connectivity to ATM



- Connect via PCI bus
  - Interface/translate to GALNET protocol
- Alternatively, use high speed CPU local bus with standard ATM chips
  - Communicate via System Controller

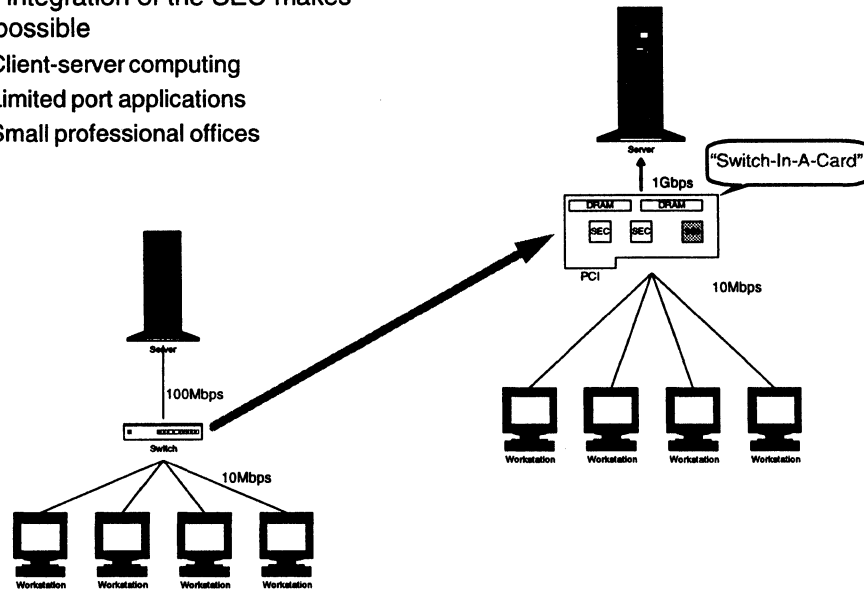


APRIL 1996

14

## Switch In A Card

- High integration of the SEC makes this possible
  - Client-server computing
  - Limited port applications
  - Small professional offices



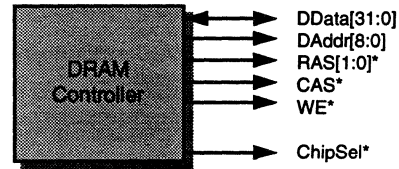
## Architectural Overview

- Interfaces
  - DRAM interface
  - PCI interface
  - Serial interfaces
  - LED interface
- Switching & address recognition engines
- Packet forwarding buffers & queues
- Management support
  - Intervention mode
  - Statistics counters
  - RMON
  - Sniffer
  - Spanning tree
- Watchdog timer
- Default configuration

## DRAM Controller



- Direct 32-bit interface to 1 or 2MByte DRAM
- Support for 60ns standard or EDO
  - EDO shortens latency between packets
  - Auto-configuring interface timing
- DRAM stores 8K address table
- DRAM used for storing packets
  - Fixed receive buffer size of 1.5KBytes
  - Supports 1024 buffers (2MB) or 384 buffers (1MB)
  - Buffers dynamically allocated to the ports and the PCI bus
- ChipSel\* signal used for interfacing to an external FIFO
  - Store destination, source & byte count observed on DData[31:0] for RMON



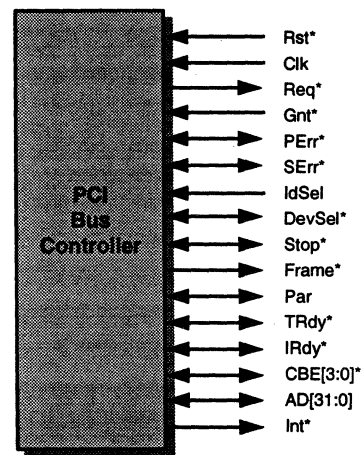
APRIL 1996

17

## PCI Interface



- 1Gbps bandwidth
  - 133MBytes/sec Bandwidth @33Mhz
- Provides expansion and connectivity
  - GALNET protocol operates on the PCI
  - Up to 32 SEC Devices (256 Ports)
  - Up to 6 SEC Devices Without PCI-PCI Bridges
  - CPU Connection for Management
  - High Speed LAN (100Mbps, ATM, etc.)
- PCI Master and Slave Functionality
- Supports version 2.1 specification
- Supports clock frequencies of 25-33MHz
- External reset
- Full parity support



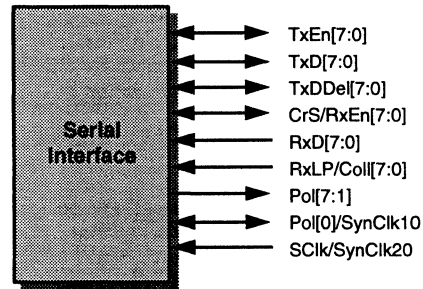
APRIL 1996

18

## Tx and Rx Serial Interfaces



- 8 Ethernet ports
  - ➔ MAC, Manchester Encoder/Decoder, Link integrity, Auto-Polarity, Dual 32-Byte FIFOs, 7 LEDs
- Compliance with Ethernet and 802.3
  - ➔ Production-proven MAC
- Full-duplex support (20Mbps)
- <\$2.00/port in external drivers/filters in 10baseT
- Supports different PHY options, each can be set individually per port
  - ➔ 10base-T
  - ➔ 10base-F
  - ➔ AUI
  - ➔ NRZ Synch.
- All digital logic, no analog elements
- 80MHz serial clock



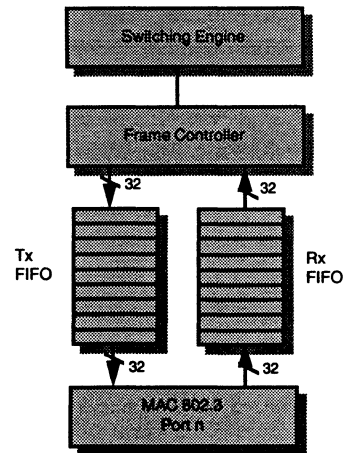
APRIL 1996

19

## Tx and Rx FIFOs



- Used to buffer incoming and outgoing packets
  - ➔ Fully bidirectional and independent
  - ➔ 32-bytes for receive & 32-bytes for transmit
  - ➔ One pair per port
- Interfaces between the Frame Controller/Switching Engine and the MACs



APRIL 1996

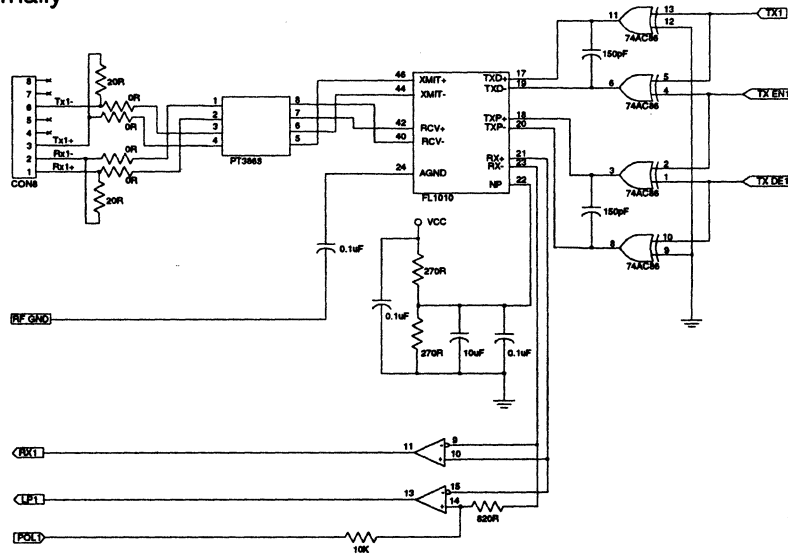
20



## 10-Base-T Implementation



- Low cost components required externally



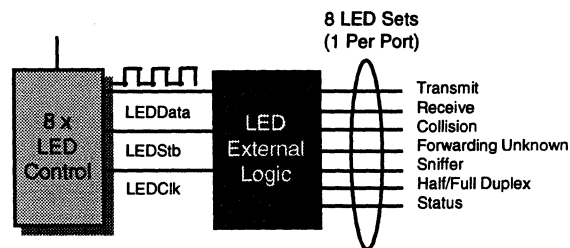
APRIL 1996

21

## LED Interface



- Serial shift register output contains 7 LED indicators per port
- LED indicators
  - ➔ 7 Data (one set per port)
    - ▲ Receive, Transmit, Collision, Forward Unknown, Sniffer, Half/Full Duplex, Link Status
  - ➔ Strobe
    - ▲ Indicates start of valid data
  - ➔ Clock
    - ▲ Used to clock the serial data out



APRIL 1996

22

## **Switching Engine**



- Store and Forward architecture
- Patent pending technology
- Provides the switching function between:
  - ➔ 8 Ethernet ports and PCI bus
- Guarantees full wire speed for all ports with NO latencies
- Supports the management features
  - ➔ Spanning tree, station-to-station connectivity matrix, etc
- Supports the powerful Intervention Mode
- Allocates the buffer sizes

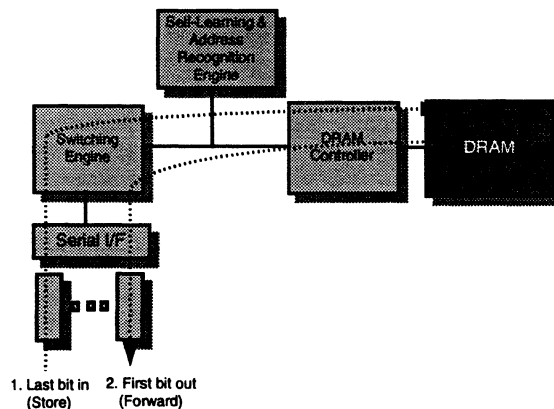
APRIL 1996

23

## **Switching Engine Performance**



- High aggregate throughput
  - ➔ > 650K Unicast packets/sec (using 3 SECs)
  - ➔ > 90K Multicast packets/sec (using 3 SECs)
- Low last bit-to-first bit delay
  - ➔ Ports at the same SEC:
    - ▲ Short packet: 3us; long packet: 3us
  - ➔ Ports at different SECs:
    - ▲ 2 SECs - short packet: 4us; long packet: 18us
    - ▲ 4 SECs - short packet: 8us; long packet: 30us

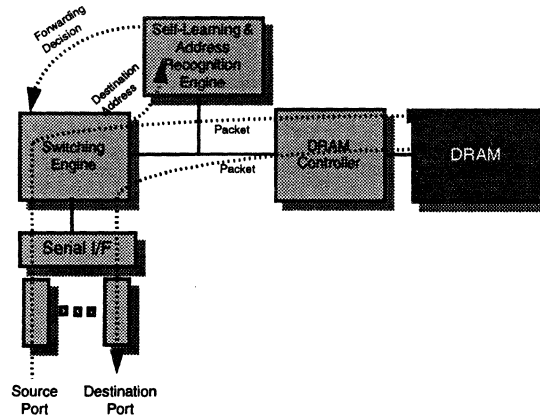


APRIL 1996

24

## Address Recognition

- The Address Recognition Engine looks for the destination address in the incoming packet
  - ➔ A match will occur if the same destination address has been received before
  - ➔ The associated data containing the port address directs the packet to the specified port
- Unicast addresses
  - ➔ Port # and Device # match
    - ▲ Packet is discarded
  - ➔ Port # is different, Device# matches
    - ▲ Packet is forwarded to the proper Port # in the same device
  - ➔ Device # is different
    - ▲ Packet is forwarded to another SEC via PCI bus



APRIL 1996

25

## Address Recognition

- Intelligent address recognition
- Supports up to 8K MAC addresses
- Full speed frame forwarding
- Self learning mechanism
  - ➔ CPU optional
- Address Table stored in DRAM
  - ➔ Very cost effective
- Additional bits for:
  - ➔ Device number (Dev#, 5-bits)
  - ➔ Port number (Port#, 3-bits)
  - ➔ Valid (V)
  - ➔ Skip (Sk)
  - ➔ Aging (A)
  - ➔ Static address (St)
  - ➔ Multiple (M)
  - ➔ Intervention for DA (Id)
  - ➔ Intervention for SA (Is)
- Broadcast storm rate filtering

	63	62	61	60	59	58-56	55-51	50	3 2 1 0		
	Is	Id	M	St	R	Port#	Dev#	Address [47:0]	A	Sk	V
1											
8K											

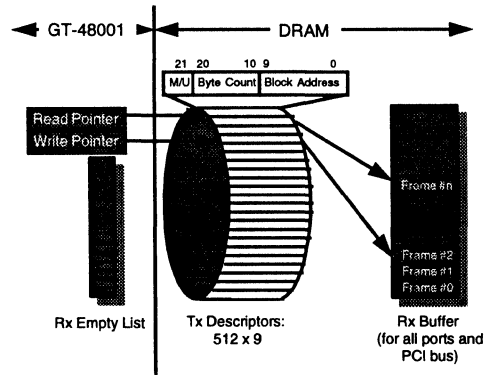
APRIL 1996

26

## Buffers and Queues



- 9 transmit queues
  - 8 ports and PCI bus
  - Both transmit queues and receive buffers are held in DRAM
- Tx descriptors
  - 9 descriptor rings, each containing 512 descriptors
    - ▲ Size is 32-bits, includes Block Address/1536, Byte Count, Packet Type (Multicast or Unicast)
- Read/Write Pointers
  - 9 pairs of pointers to the transmit descriptors
- Rx Buffer
  - Common for all ports
  - 384 blocks (1M DRAM) or 1024 blocks (2M DRAM) of 1536 bytes each
- Rx Empty List
  - Each block (384 or 1024) contains a bit which specifies Empty or Occupied



APRIL 1996

27

## GALNET Protocol



- 5 simple messages on the PCI
  - Efficient mechanism
    - Takes advantage of PCI's high bandwidth with little overhead
  - Enables communication between SECs
    - With or without CPU
  - Enables communication with other GALNET family members
- 1. 'new\_address' message
    - Message between GT-48001s or GT-48001 and the CPU
    - Indicates a new address
    - Also used by the CPU to update the Address Table

<b>Address</b>	Target device number 'new_address' message identifier
<b>Data</b>	
Data 0	MAC address [19:47] Aging Skip Valid
Data 1	Unknown/new address Multiple Static Address Port# Device # MAC address [0:18]
Data 2	Intervention mode for DA Intervention mode for SA

APRIL 1996

28

## GALNET Protocol (cont'd)



- 2. `buffer\_request' message
  - A message from the source device to the target device to request a buffer
- 3. `start\_of\_packet' message
  - Message from the target device to the source device which contains the empty buffer

<b>Address</b>	Target device number 'buffer_request' message identifier Source device number
<b>Data</b>	
Data 0	Sniffer type Unknown message Source port # Target port # Multicast/Unicast Byte count Source buffer address

<b>Address</b>	Source device number 'start_of_packet' message identifier
<b>Data</b>	
Data 0	Sniffer type Target port # Multicast/Unicast Byte count Target buffer address
Data 1	Source port # Source buffer address Target device number

APRIL 1996

29

## GALNET Protocol (cont'd)



- 4. `packet\_transfer' message
  - Burst of 8 32-bit words from the source device to the target device which contains the packet
- 5. `end\_of\_packet' message
  - Message from the source device to the target device which indicates end of packet

<b>Address</b>	Target device number 'packet_transfer' message identifier DRAM location
<b>Data 0</b>	
----	
----	Data 0
----	----
<b>Data 7</b>	
----	
----	Data 7

<b>Address</b>	Target device number 'end_of_packet' message identifier
<b>Data 0</b>	Unknown packet Target port # Multicast/Unicast Byte count Target buffer address

APRIL 1996

30

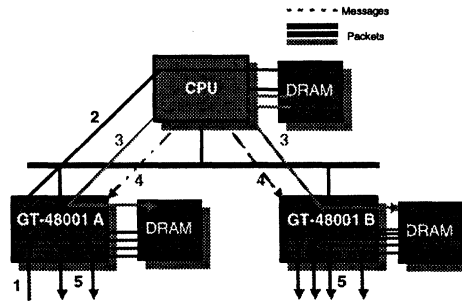
## Management: Multicast Intervention Mode



### ■ Intervention mode for Multicast packets

- ➔ All multicast packets forwarded to CPU memory
- ➔ CPU decides which ports to send the packets to
- ➔ Sequence

- ▲ 1. Incoming packet is received & stored in GT-48001 (A) DRAM
- ▲ 2. GT-48001(A) transfers packet to CPU main memory
- ▲ 3. CPU transfers packet to the selected devices GT-48001 (A,B)
- ▲ 4. When finished sending packets, CPU sends 'end\_of\_packet' message to tag selected ports
- ▲ 5. Packet is transmitted on selected ports



APRIL 1996

31

## Management: Unicast Intervention Mode

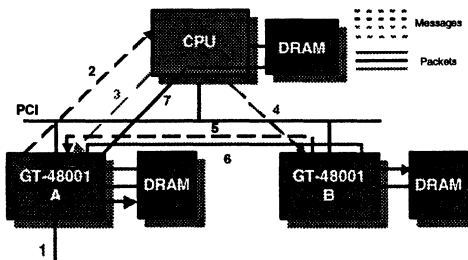


### ■ Intervention Mode for Unicast packets

- ➔ Optional per MAC source or destination address or both

### ■ Sequence

- ▲ 1. Unicast packet is received & stored in GT48001 (A) memory
- ▲ 2. If an intervention bit is set, GT48001(A) sends a 'buffer\_request' to the CPU, including the source port & the destination port/device
- ▲ CPU alternative 'a'
  - ➔ 3. Discard the packet
- ▲ CPU alternative 'b'
  - ➔ 4. Signal to forward the packet to the destination device.
  - ➔ 5. Buffer allocated at GT48001(B), which sends a 'start\_of\_packet' to GT48001(A)
  - ➔ 6. GT48001(A) sends packet to GT48001(B)
- ▲ CPU alternative 'c'
  - ➔ 7. Take the packet and modify it. GT48001(A) sends the packet to the CPU main memory.



APRIL 1996

32

## Management: Statistics Counters



- Repeater MIB & PCI counters
  - MIB: 15, 32-bit wrap-around counters per port
    - ▲ Bytes received
    - ▲ Multicast bytes received
    - ▲ Broadcast bytes received
    - ▲ Bytes sent
    - ▲ Frames received
    - ▲ Multicast frames received
    - ▲ Broadcast frames received
    - ▲ Frames sent
    - ▲ Collision
    - ▲ Late collision
    - ▲ CRC + alignment
    - ▲ Jabber
    - ▲ Frame too short
    - ▲ Frame too long
    - ▲ Bad (CRC error, frame too long) bytes received
  - Global PCI traffic 32-bit counters for ALL ports
    - ▲ PCI frames received
    - ▲ PCI frames sent

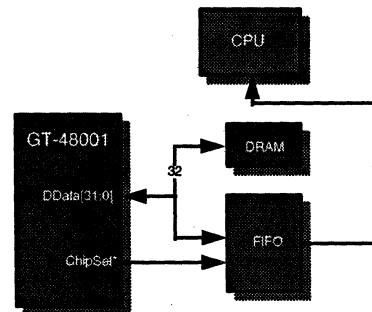
APRIL 1996

33

## Management: RMON



- Station-to-station connectivity matrix
  - Records destination address, source address, and byte count for all forwarded packets
  - Used for RMON support
  - Stored in an external FIFO
- ChipSel\* signal used for interfacing to an external FIFO
  - Store destination, source & byte count observed on DData[31:0] for RMON



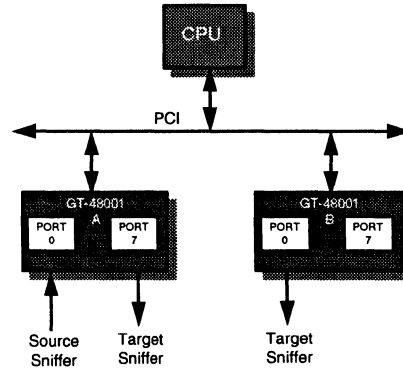
APRIL 1996

34

## Management: Sniffer Mode



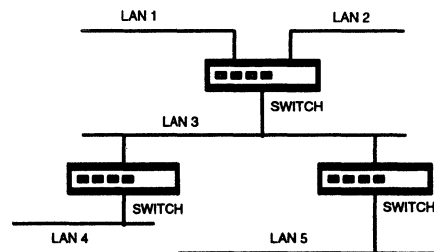
- Monitoring (Sniffer) mode
  - Target can be the ports within the same SEC or ports in another SEC
  - One port is set to monitor
  - All traffic from other ports and/or SECs is received by the Sniffer port
  - Target Sniffer is written to the CPU and the Sniffer Numbers register
  - LED active on Sniffer port



## Management: Spanning Tree



- Spanning Tree support
  - GT-48001 provides the hardware assistance to implement the spanning tree algorithm
  - CPU executes the algorithm



SpanEn (Global)	SpanEn (Port)	Logic State	Remarks
0	x	Port Enable	No Spanning Tree
1	1	Blocking Listening Learning*	1. Transfer BPDU to CPU 2. All Rx/Tx packets rejected 3. Accept BPDU messages from CPU 4. No address learning in this stage
1	0	Forward	1. Transfer BPDU to CPU 2. Accept all packets 3. Address learning

\* During the state called Learning per the Spanning Tree protocol, the GT-48001 actually does not learn. It starts learning during the Forward state.



## Watchdog Timer



- Used for every transmit queue
- Used to empty packets from the queue when the timer expires
- When timer expires:
  - ➔ GT48001 clears the used blocks & sends an interrupt (Int\*) to the CPU
  - ➔ Default value 60ms
  - ➔ Range is 10 - 160ms

0x0	0000	Illegal
0x0	0001	10ms
0x0	0010	20
0x0	0011	30
0x0	0100	40
0x0	0101	50
0x0	0110	60 (DEFAULT 0x6)
0x1	0000	160 (MAX 0x10)

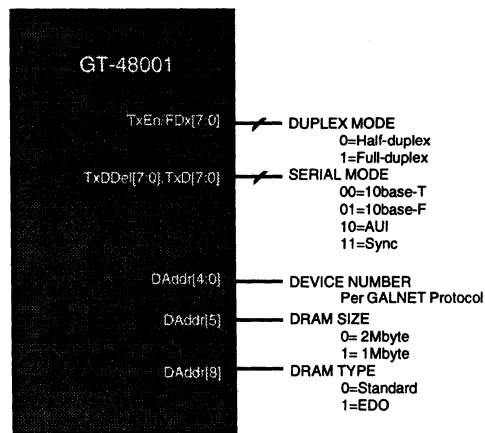
APRIL 1996

37

## Configuration at Reset



- SEC acquires basic system knowledge during initialization
  - ➔ Certain pins must have pull-up or pull-down resistors
  - ➔ SEC samples them at reset
- Allows operation without CPU
- Basic parameters set
  - ➔ Duplex mode per port
  - ➔ Serial mode per port
  - ➔ Device number per GALNET protocol
  - ➔ DRAM size
  - ➔ DRAM type



APRIL 1996

38

## **Conclusion**

- The GALNET architecture offers a flexible high performance path to design advanced internetworking equipment
- Use the PCI as a robust and expandable backbone
  - 32-bit, 33MHz
  - 64-bit and 66-MHz in the future
- The GT-48001 SEC addresses with excellent price/performance the large 10Mbps switched market
- Low end to high end equipment may be built
  - Unmanaged low end (no CPU) to heavily managed high end (advanced RISC CPU)

## USING FPGAs FOR HIGH-PERFORMANCE PCI

James D. Joseph  
Actel Corporation  
955 East Arques Avenue  
Sunnyvale, CA 94086-4533  
e-mail: [jjoseph@actel.com](mailto:jjoseph@actel.com)

### ***ABSTRACT***

Supporting PCI interfaces in programmable logic is now mandatory. Even systems not currently using PCI add a PCI interface "just in case." However, merely supporting the PCI protocol is inadequate, in our opinion. Unleashing the full potential of PCI system bandwidth requires support for zero-wait-state operation.

The Actel Act-3 FPGA family provides high density, high speed, and compact registered I/O cells. This combination allows implementation of high-performance target, master, and bridge functions in FPGAs. Designers are able to use these functions in conjunction with on-chip programmable logic to generate cost-effective PCI system solutions.

Actel's target, master, and PCI-PCI bridge are all oriented toward high-performance data transfer while still meeting the stringent PCI I/O drive requirements. The result is an effective system-oriented solution for 5-volt, 33-Mhz PCI. The target macro is compatible with a number of synthesis and simulation paths including Synplicity, Model Technologies, Exemplar, and Synopys.

### ***THE PROBLEMS***

The PCI bus standard was created to improve data transfer rates among peripherals, main memory, and CPUs. The major competition at the time the standard was created was the VESA local bus, and most VL controllers did not support bursting. In addition, early PCI chipsets had mediocre transfer rates, so even a target or master that could support zero-wait-state bursting or fast leadoff cycles was often strangled by an inefficient chipset.

All that has now changed. As familiarity with the PCI bus has grown, so have the systems ready to take advantage of its high performance. Higher speeds in FPGAs make the job of designing a target, master, or bridge much easier. Zero-wait-state bursting is available in chipsets supporting PCI interfaces. Better arbitration logic also allows faster PCI leadoff cycles. This high performance also must be achieved with the convenience of a high-level model. The model must be available in a VHDL and/or Verilog format compatible with today's synthesis tools from multiple vendors.

The third issue is flexibility. The designer wants to customize the models to meet application-specific requirements with a minimum of fuss.

The final major issue is I/O compliance. The PCI 5v specification, defined in terms of a voltage-current map, demands both high drive and high slew rate. 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.

The discussion which follows does not include configuration cycles. These, of course, are handled by the Actel PCI macros, but the speed of configuration cycles is not an issue in high-performance systems. We begin with a discussion of requirements for a PCI target macro in the Actel ACT3 FPGA family.

## ***ACT3 FEATURES***

The ACT3 family offers a collection of features well-suited to 33-Mhz PCI design. These include the following:

- Acceptable speed ( $t_{CO} = 7.5$  ns)
- Fast, low-skew clock networks
- High number of user I/Os (up to 228)

In addition, the ACT3 family has been extended to provide I/O buffers which are fully PCI compliant.

## ***PCI TARGET REQUIREMENTS/GOALS***

The example chosen is a peripheral controller having both I/O and memory requirements. The vast majority of target designs support either memory cycles alone or both memory and I/O cycles. The I/O space is composed of 64 contiguous bytes with a memory space of 16 megabytes.

The design goal was implementation in generic VHDL where possible. Key features include the following:

- Memory Read/Write Bursting Support
- :1:1:1 Sustained Performance Possible
- Fast Back-to-Back Transaction Support

In addition, we wanted a modular structure that would accommodate a wide range of target designs. This required not only easily-modified code segments but also a flexible back-end interface.

## ***SUPPORTED COMMANDS***

The supported command types for the target macro are briefly described below.

### **I/O Read and I/O Write**

These commands read data from/write data to the target's I/O address space. The macro as defined has a 64 byte I/O address space.

### **Memory Read and Memory Write**

These commands read data from/write data to the target's memory-mapped address space. The memory as defined in the macro is 16 megabytes which can be located anywhere in 32-bit address space.

### **Configuration Read and Write**

These commands access the configuration space of the target. A PCI target, master, or bridge MUST implement these commands. All required configuration registers are supported in the Actel target macro. In addition, the target supports memory and I/O base address registers.

### **Memory Read Multiple**

This command is similar to a memory read command, but it indicates that the master may intend to fetch more than one cache line. The memory controller should continue to fetch data so long as FRAMEn is asserted.

## ***THE BACK-END INTERFACE***

A number of signals are provided for maximum flexibility in interfacing the target's PCI bus to the user's 32-bit external logic. Naturally, address, data, and byte-enables are available to/provided from the user's logic. In addition, the target provides signals which allow a variety of data flow control for the user application depending on required performance and acceptable complexity. Both read and write functions can operate in the following transfer modes:

- Burst mode where a DWORD is transferred every cycle (some limitations exist for read commands)
- Fast mode where a DWORD can be transferred every other cycle (all configuration transactions are in the Fast mode)
- Handshake mode for lower-performance peripherals (maximum of one DWORD every 3rd cycle)

In addition, the target can throttle the speed of the master in either burst or fast mode to allow occasional reductions in transfer rates (e.g., memory refresh).

The key signals that support this flexible interface are:

**TARGET\_ACTIVEN** - Active low signal indicating that the target has claimed the bus and is active. This signal will remain active as long as the PCI **DEVSELn** signal is active.

**LAST\_CYCLE** - Active high signal indicating that the PCI bus transaction is in its last phase of transfer.

**DATAPHASE** - Active high signal indicating that a data phase (**IRDYn** and **TRDYn** both low) occurred on the previous cycle. **DATAPHASE** is used as a handshake mechanism for both read and write cycles.

**MEM\_READ** - Active high output indicating a read from the target's memory space has been detected.

**IO\_READ** - Active high output indicating a read from the target's I/O space has been detected.

**READ\_OK** - Active high input indicating that the user defined function is ready to transfer data to the PCI bus.

**NEXT\_READ\_OK** - Active high input indicating that a burst is possible because more than one DWORD is available to transfer at the 33 Mhz rate.

**MEM\_WRITE** - Active high signal indicating a write to the target's memory space has been detected.

**IO\_READ** - Active high signal indicating a read from the target's I/O space has been detected.

**WRITE\_OK** - Active high input indicating that the user function is ready to receive data from the PCI bus.

**NEXT\_WRITE\_OK** - Active high input indicating that a burst is possible because more than one DWORD is available to transfer at the 33 Mhz rate.

**BUSY** - Active high input indicating the user-defined function cannot currently respond to a PCI request.

**ERROR** - Active high input indicating that an error has occurred and will initiate a Target Abort cycle.

**FATAL\_ERROR** - Active high input indicating that an error has occurred and will initiate a Target Abort cycle and assert the **SERRn** signal on the PCI bus.

Figures 1 and 2 illustrates the timing in burst mode and fast mode, respectively, for the Actel PCI target macro.

## ***EXTENDING MACRO PERFORMANCE***

All of these goals were met in the behavioral version of the target macro. Nevertheless, we chose to improve performance further by instantiating a high-speed Actel **ACTgen** counter macro. The result was a smaller AND faster design.

Table 1 summarizes the device usage in the Actel 1460A FPGA. Two different versions are illustrated. The first is a target that responds to both memory and I/O accesses, while the second case is a memory-only target.

Table 1  
FPGA Usage for Target Macros

- Memory + I/O
  - Combinational Cells: 277
  - Sequential Cells: 163
  - Total Cells: 440 of 848 (53%)
- Memory Only
  - Combinational Cells: 208
  - Sequential Cells: 122
  - Total Cells: 330 of 848 (39%)

### **CONCLUSION**

Zero-wait-state performance for PCI target, master, and bridge macros can be achieved in FPGAs using high-level design and synthesis techniques with a combination of high-performance parts, careful design, and Actel's ACTgen macros. The designs presented here achieve full 5-volt PCI compliance at 33 MHz.

### **REFERENCES**

PCI Special Interest Group, 1995. PCI Local Bus Specification, Revision 2.1, 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.



## AUTHOR INDEX

Alba, Manuel	500	Kolment, Raymond	13
Anderson, Thomas L.	86	Kupnicki, Richard A.	366
Apte, Dave	479	Lee, Edwin	17, 178
Armstrong, Ross L.	93	Lipman, Jim	464
Arunarathi, Venkatesh	488	Mahmood, Aamer	233
Autor, Jeff	51	Mansharamani, Kamal	101
Becker, Mike	61	McCook, Donald F.	243
Blackledge, Larry	302	McGowen, Michael	234
Blau, Bob	161	Medeiros, Jim	312
Bronson, Mark	394	Mourn, Richard	262
Burk, Richard J.	22, 191	Nayak, Harish	1
Carter, Jack	160	Nygaard, Thomas	390
Chisvin, Larry	463	Ohr, Stephan	285
Cohen, Frances	452	Pavlat, Joe	409
Cooper, Steve	284	Rao, K. K.	255
Craven, Tim	443	Rathnam, Selliah	275
Cruz, Claude A.	44	Rhoden, Desi	268
Dahlgren, Kent	171	Richardson, Tracy	71
Dingee, Don	399	Ridgeway, David	199
Evans, Dave	465	Riley, Dwight D.	109
Fitzgerald, Jim	187	Ryan, Arthur	195
Forbes, Bert	186	Salameh, Mike	200
Ganesan, Subbu	494	Sawant, Sanjay	485
Garrett, Billy	225	Schapel, Frank	274
Gildersleeve, Gary	8	Schiefer, Harold	381
Glaskowsky, Peter N.	201	Serra, Fernando	358
Goodrum, Alan	51	Sgro, Joseph A.	119
Goodwin, Bob	214	Sheberman, Tony	7
Greggain, Lance	381	Small, Brian	343
Hady, Frank	26	Stearns, Margit E.	211
Hanna, Steven	381	Thomas, Clyde	12
Hawes, Adge	328	Thorsteinson, Tom	353
Hoff, James F.	347	Trevett, Neil	215
Hosking, Rodger H.	154	Venkat, Giri	283
Isenstein, Barry	161	White, George P.	286
Jensen, Scott	261	Won, Martin	494
Joseph, James D.	520	Wong, Jacques	86
Kasiwal, Manish	160	Wong, Leo K.	334, 494
Kelsey, Jim	454	Yao, Yong	440
Knecht, Mark W.	86	Yeung, Ernest	381





# KEYWORD INDEX

- 3D graphics 215
- Accelerators 119, 215
  - Alpha 394
  - ASICs 187, 195, 234, 465, 488
  - ATM 201, 494, 500
- Backplanes 17, 93, 409
- BIOS 51, 211, 452, 454
- Board computers 394, 399
- Boot 255, 452
- Bridges 26, 44, 51, 71, 86, 93, 101, 285, 286, 394, 399, 462, 463, 520
- Broadcasting 366
- Buffering 71
- Burst mode 26, 494
- Bus analyzers 390
- Bus arbitration 86, 211, 262
- Bus interface 8
- Bus interface chips 399
- Bus interface models 479
- Bus mastering 109, 187, 191, 233, 347, 462
- Bus-to-bus connections 394
  - C-bus 286
  - Cache 286
  - CAD 464, 494
  - CardBus 8, 44
  - Chipsets 26
  - CHRP 399
  - Clusters 119
- Communications 13, 353
  - CompactPCI 17, 178, 284, 312, 390, 409
- Compatibility 101
- Compliance 520
- Computer-aided design 465, 479, 485, 488
- Computer-telephony integration 13
- Configuration 51
- Control systems 17
  - Cores 465
  - CPLDs 334, 358, 494
- Data acquisition 22, 154, 187, 195
- Data acquisition systems 191
  - Dataflow 171
- Debugging 390
- Delayed transactions 71
- Design kits 347, 358
- Device controllers 452
- Digital signal processing 119, 154, 160
  - Disk arrays 255
  - Disk controllers 255, 261, 262
- Distributed DMA 109
  - DMA 1, 86, 109, 187, 358
- Docking 1, 44, 454
- Efficiency 26
- Electrical compliance 343
- Embedded systems 161, 200, 312, 409
  - Emulation 485
  - Ethernet 500
  - Fast Ethernet 500
  - Fire wire 302
  - FPGAs 187, 191, 343, 347, 358, 520
- Full-motion video 214
- Frame grabbers 358
  - GPIO 195
- Graphics 214, 215, 225, 268, 283
  - HDLs 334, 347, 358, 479, 488
- High-level design 520
  - HIPPI 234
- Hot swapping 233
  - 1960 200
  - IEEE 1394 302
- I/O boards 160, 394
- I/O interfaces 262
- Image processing 22, 119, 275, 358, 381
- Industrial applications 12, 13
  - Industrial control 17, 93, 178, 243, 284, 409
- Instrumentation 195
  - Interface chips 187, 191, 200, 334, 343, 347, 358
- Interoperability 494
  - Interrupts 1, 8, 51, 86, 93, 211
  - ISA 1, 22, 86, 101, 109, 187, 312
- Isochronous data 302
  - JPEG 214, 358, 366, 381
- LANs 500
- Latency 233, 302
- Legacy systems 109, 312
- Machine vision 358
- Maintenance 443

Marketing	440
Mechanical design	443
Medical applications	22
Memory interface	191, 225, 268
Mezzanine bus	154
Minicard	7
Mobile applications	7, 454
Mobile computing	44, 243
Mobile systems	1
Models	465, 479, 488
Motherboards	211, 268, 399, 440, 443
MPEG	274, 275, 283, 353, 366, 463
Multicomputers	161
Multifunction devices	211
Multimedia	44, 101, 214, 215, 225, 274, 275, 283, 302, 358, 366, 462, 463
Multiprocessors	119, 286
Network adapters	233
Networking	201, 233, 234, 328, 353, 494, 500
Notebook computers	454
Packet switching	13
Parallel processors	119
PC cards	7, 8, 44
Performance analysis	26, 201, 488
Personal computers	440, 443
PICMG	13, 17, 93, 178, 284, 409
PLDs	343, 358
Plug and play	452, 454, 463
PMC	154, 161, 234, 390
Portable computers	7
Portable systems	243
PowerPC	61, 399
Power-up	452
Product development	465
Programmable logic	520

RACEway	161
RAID	255, 261
RAM	225, 268
Rambus	225
Real-time systems	160
RISC	274
RISC processors	234
Scaling	171, 381
SCSI	255, 262, 328
Semicustom logic	334, 358
Serial interfaces	261, 328
Serial storage architecture	285
Servers	51, 101, 171, 201, 261, 262, 286, 328
Shared memory	171
Simulation	479, 488
STD 32	312
SSA	328
Switching	161, 171
T-1	353
Telecommunications	22, 409
Test and measurement	195
Test equipment	390
Testing	465, 485
Timing requirements	334, 343
UMA	268
Verification	485
VHDL	494
Video applications	353
Video compression	358, 381
Video processing	171, 214, 274, 275, 358, 366, 381, 462
Vision applications	358
VLIW	275
VME	154, 161, 178, 195, 394
VXI	195
WANs	353
Workstations	201, 215
Zoomed Video (ZV) port	8

## PARTICIPANTS LIST

**Edward Agis**, Texas Instruments Inc., Semiconductor Group, Hwy 75 South, MS 835, Sherman, TX, 75091, USA, 903-868-7280

**Barbara Aichinger**, FuturePlus Systems Corp, 36 Olde English Rd., Bedford, NH, 03110, USA, 603-471-2734

**Manuel Alba**, Galileo Technology, 1735 N. First St. #308, San Jose, CA, 95112, USA, 408-451-1400

**Ray Alderman**, VITA, 7825 E. Gelding Dr., Scottsdale, AZ, 85260-3415, USA, 602-951-8866

**Tom Anderson**, Virtual Chips Inc., 2107 N. First St., Suite 100, San Jose, CA, 95131, USA, 408-452-1600 x-219

**David M. Anderson**, Total Company Synergy, P.O. Box 1082, Lafayette, CA, 94549, USA, 510-253-0900

**Charles Anderson**, Cogent Data Technologies Inc, 175 West St, PO Box 926, Friday Harbor, WA, 98250, USA, 360-378-2929

**Warren Andrews**, RTC Magazine, 261 Concord St., West Gloucester, MA, 01930, USA, 508-283-2102

**Dave Apte**, Omniview, Inc., 100 High Tower Blvd, Suite 201, Pittsburgh, PA, 15205, USA, 412-788-9492

**Rod Archer**, Award Software, 777 E. Middlefield Rd, Mountain View, CA, 94043-4023, USA, 415-968-4433 x266

**Ross Armstrong**, Digital Equipment, Mosshill Industrial Estate, Ayr, KA6 6BE, Scotland, 011441292 266955

**Venkatesh Arunarthi**, Sand Microelectronics, 1630 Oakland Road, A-103, San Jose, CA, 95131, USA, 408-441-7138

**Jeff Autor**, Compaq Computer Corp., MS 090703, PO Box 692000, Houston, TX, 77269, USA, 713-518-8934

**Pradeep Bardia**, Texas Instruments, Semiconductor Group, MS 835, PO Box 84, Highway 75 South, Sherman, TX, 75091, USA, 903-868-5110

**Mike Becker**, Motorola, MD OE512, 9737 Great Hills Trail, Austin, TX, 78759, USA, 512-795-7255

**Jim Beedle**, In-Stat, Inc., 7418 E. Helm Dr., Scottsdale, AZ, 85260, USA, 602-483-4463

**Kanti Bhaduthmal**, Sun Microsystems, SPARC Technology Business, 2550 Garcia Ave, Mountain View, CA, 94043, USA, 408-774-8006

**Roger Billings**, WideBand Corp., 26900 East Pink Hill Road, Independence, MO, 64057, USA, 816-220-3000

**Larry Blackledge**, Texas Instruments Inc., Data Transmission Products, 8505 Forest Lane, MS 8710, Dallas, TX, 75243, USA, 214-997-3603

**Keith Bladen**, Intel Corp., PCI Components Division, 1900 Prairie City Rd., MS FM5-62, Folsom, CA, 95630, USA, 916-356-8064

**Robert Blau**, Mercury Computer Systems, 199 Riverneck Rd., Chelmsford, MA, 01824, USA, 508-256-1300

**Jag Bolaria**, Intel Corp., PCI Components Division, 1900 Prairie City Rd, Folsom, CA, 95630, USA, 916-356-8064

**Chris Borghi**, FirePower Systems, 190 Independence Dr, Menlo Park, CA, 94025, USA, 415-462-6237

**Mark Bronson**, c/o Barbara Patterson, Patterson Assoc. (Aeon Systems), 4205 E. Desert Crest Dr., Paradise Valley, AZ, 85253, USA, 505-628-9120

**Mark Brown**, Intel Corp., Enterprise Computing I/O, 5000 W Chandler Blvd, Chandler, AZ, 85226, USA, 602-554-3864

**Marshall Brumer**, Microsoft, One Microsoft Way, Redmond, WA, 98052, USA, 206-936-5840

**Richard Burk**, Data Translation, 100 Locke Dr, Marlboro, MA, 01752, USA, 508-481-3700

**Shelagh Callahan**, PC2 Consulting, 3280 SW 170th Ave, #1608, Beaverton, OR, 97006, USA, 503-264-8440

**Jack Carter**, Sonitech International, 14 Mica Lane, Wellesley, MA, 02181, USA, 617-235-6824

**Steve Chen**, Chen Systems, 1414 W. Hamilton Ave, Eau Claire, WI, 54701, USA, 715-833-7067

**Larry Chisvin**, S3, Inc., 2770 San Tomas Expy, Santa Clara, CA, 95051, USA, 408-980-5401 x-3062

**Kevin Christiansen**, Apple Computer Inc., MDS, 20525 Mariani Ave, MS 60-IO, Cupertino, CA, 95014, USA, 408-974-0308

**Frances Cohen**, Phoenix Technologies Inc., 2575 McCabe Way, Irvine, CA, 92714, USA, 714-440-8323

**Bernard Cole**, EE Times, 2700 Woodlands Village Blvd, Suite 300-418, Flagstaff, AZ, 86001, USA, 602-773-7873

**Todd Comins**, Digital Semiconductor, Digital Equipment Corp., 77 Reed Rd., Hudson, MA, 01749-2895, USA, 508-568-5179

**Steve Cooper**, I-Bus, 9596 Chesapeake Dr., San Diego, CA, 92123, USA, 619-974-8424

**Patrick Correia**, Intel Corp., PCI Components Division, 1900 Prairie City Rd, Folsom, CA, 95630, USA, 916-356-8064

**Tim Craven**, Intel Corp., 5200 NE Elam Young Pkwy, Mail Stop HF3-89, Hillsboro, OR, 97124, USA, 503-696-5805

**Ian Crayford**, AMD, One AMD Place, PO Box 3453, Sunnyvale, CA, 94088, USA,  
408-749-5449

**Claude Cruz**, National Semiconductor Corp., 333 Western Ave., M/S 10-26, South Portland,  
ME, 04106, USA, 207-775-8318

**Robert Cutler**, AMP Inc., 24736 W. Saddlepeak Rd, Malibu, CA, 90265, USA, 310-456-5325

**Kent Dahlgren**, I-Cube, 2328-C Walsh Ave, Santa Clara, CA, 95051, USA, 408-986-1077

**Rob Davidson**, Ziatech Corp., 1050 Southwood Dr., San Luis Obispo, CA, 93401, USA,  
805-541-0488

**Alan Deikman**, Znyx Corp., 48501 Warm Springs Bl, Suite 107, Fremont, CA, 94539, USA,  
510-249-0800

**Jim Detar**, Electronic News, West Coast Office, 1205 Landings Dr., Mountain View, CA,  
94043, USA, 415-691-1656

**David Dickens**, PCIbus Solutions, Texas Instruments, PO Box 84, Sherman, TX, 75090,  
USA, 903-868-7594

**Joseph DiMartino**, IBM PC Company, 3039 Cornwallis Rd., MS 201-K103B-91B,  
Research Triangle Park, NC, 37709, USA, 919-543-9795

**Don Dingee**, Motorola Computer Group, C/O JERRY GIPPER, 2900 S. Diablo Way,  
Tempe, AZ, 85282, USA, 602-438-3025

**Thomas Dippon**, Hewlett-Packard, Herrenberger Strabe 130, PO Box 14 30,  
Boeblingen, 71034, Germany, (49) 7031-14-3973

**Michael Eckley**, Interphase, 13800 Senlac, Dallas, TX, 75234, USA, 214-654-5325

**Tim Elsmore**, Vigna, 10052 Mesa Ridge Ct., San Diego, CA, 92121, USA, 619-597-7080

**Ross Ely**, Apple Computer Inc., One Infinite Loop, MS:306-4PM, Cupertino, CA, 95014,  
USA, 408-974-1846

**Tim Erjavec**, Chips and Technologies, Inc., 2950 Zanker Rd, San Jose, CA, 95134, USA,  
408-434-0601 x-4354

**Dave Evans**, Technical Data Freeway, 9019 Twin Trails Ct, Suite 201, San Diego, CA, 92129,  
USA, 619-538-0825

**Wayne Fischer**, Force Computers Inc, 2001 Logic Dr, San Jose, CA, 95124, USA,  
408-369-6260

**Jim Fitzgerald**, Keithley MetraByte, 440 Myles Standish Blvd, Taunton, MA, 02780, USA,  
508-821-1717

**Bert Forbes**, Ziatech Corp., 1050 Southwood Dr., San Luis Obispo, CA, 93401, USA,  
805-541-0488

**Billy Garrett**, Rambus, Inc., 2465 Latham St., Mountain View, CA, 94040, USA,  
415-903-3800

**Gary Gildersleeve**, Host Adapter Group, Cirrus Logic Inc., 3100 W. Warren Ave,  
Fremont, CA, 94538, USA, 510-623-8300

**Byron Gillespie**, Intel Corp., Embedded Processor Division, CH6-319,  
5000 W. Chandler Blvd, Chandler, AZ, 85226, USA, 602-554-2653

**Peter Glaskowsky**, Integrated Device Technology, 2972 Stender Way, Santa Clara, CA,  
95054, USA, 408-988-5636

**Ernest Godsy**, Interphase, 13800 Senlac, Dallas, TX, 75234, USA, 214-654-5325

**Mike Goodman**, Phoenix Technologies Inc., 2575 McCabe Way, Irvine, CA, 92714, USA,  
714-440-8379

**Alan Goodrum**, Compaq Computer, P.O. Box 692000, Houston, TX, 77269-2000, USA,  
713-518-8934

**Bob Goodwin**, Parallax Graphics, 2500 Condensa St., Santa Clara, CA, 95051, USA,  
408-727-2228

**Lance Greggain**, Genesis Microchip, 200 Town Centre Blvd, Suite 400, Markham, L3R 8G5,  
Canada, 905-470-2742

**Huy Ha**, Mercury Computer Corp., 199 Riverneck Rd., Chelmsford, MA, 01824, USA,  
508-256-1300

**Ken Haase**, Farallon Computing, 2470 Mariner Square Loop, Alameda, CA, 94501, USA,  
510-814-5217

**Frank Hady**, Intel Corp., M/S JF2-53, 2111 NE 25th Av, Hillsboro, OR, 97124, USA,  
503-264-8384

**Chuck Hafemann**, Imaging Technology, 55 Middlesex Turnpike, Bedford, MA, 01730, USA,  
617-275-2700

**Krista Hardie**, Electronic News, West Coast Office, 1205 Landings Dr., Mountain View, CA,  
94043, USA, 415-691-1656

**Michael Harris**, Avance Logic, 46750 Fremont Blvd., Suite 105, Fremont, CA, 94538, USA,  
510-226-9555

**Adge Hawes**, IBM, Dept. 26/19, PO Box 6, Havant, Hampshire, PO9 1SA, England,  
44-1705-486363 x-4765

**Ray Heckman**, AMD, , , , USA,

**Jim Henderson**, Innovative Integration, 31352 W. Via Colinas #101, Westlake Village, CA,  
91362, USA, 818-865-6150

**Mike Heylin**, Creative Strategies Research, 360 Ritch St, Suite 205, San Francisco, CA,  
94107, USA, 415-495-1811

**Greg Hill**, FirmWorks, 480 San Antonio Rd., Mountain View, CA, 94040, USA, 415-917-6985

**Jim Hoff**, Lucent Technology, 555 Union Blvd., Allentown, PA, 18103, USA, 610-712-5315

**Bill Holland**, IBM Networking Group, Bldg. 002, Dept. C12A, 3039 Cornwallis Rd, Research Triangle Park, NC, 27709, USA, 919-543-5444

**James Hora**, ZeitNet Inc., 5150 Great America Pkwy, Santa Clara, CA, 95054, USA, 408-562-1880 x-201

**Rodger Hosking**, c/o Barbara Patterson, Patterson & Assoc. (Pentek Corp.), 4205 E. Desert Crest Dr., Paradise Valley, AZ, 85253, USA, 201-767-7100

**Roger Hurlbert**, Trenton Terminals, 2350 Centennial Dr, Gainesville, GA, 30504, USA, 404-287-3100

**Satoshi Iida**, Motorola, Inc., Semiconductor Products Sector, 6501 William Cannon Dr., MD OE216, Austin, TX, 78735, USA, 512-891-2143

**Barry Isenstein**, Mercury Computer Systems, 199 Riverneck Dr., Chelmsford, MA, 01824, USA, 508-256-1300

**Scott Jensen**, Adaptec, C/O ADAM TRUNKEY AT ADAPTEC, 691 S. Milpitas Blvd, Milpitas, CA, 95035, USA, 408-957-6639

**Steve Johnson**, Diamond Multimedia Systems, 2880 Junction Ave., San Jose, CA, 95134, USA, 408-325-7000

**Larry Jordan**, Cypress Semiconductor, 3901 North First St., San Jose, CA, 95134, USA, 408-943-2600

**James Joseph**, Actel Corp., 5525 Evindale Dr, Suite 102, Colorado Springs, CO, 80918, USA, 719-548-4955

**Art Kahlich**, Solliday Engineering Corp., 1756 18th St, San Francisco, CA, 94107, USA, 512-933-6277

**Manish Kasliwal**, Sonitech Intl, 14 Mica Lane, Wellesley, MA, 02181, USA, 617-235-6824

**Mary Kasse**, Force Computer, 2001 Logic Dr, San Jose, CA, 95124, USA, 408-369-6260

**Mike Kelley**, Apple Computer, 1 Infinite Loop, MS 301-3K, Cupertino, CA, 95014, USA, 408-974-8535

**Jim Kelsey**, SystemSoft, 313 Speen St, Natick, MA, 01760, USA, 508-651-0088

**Joe Killian**, Killian Associates, 45 Via Del Sol, Nicasio, CA, 94946, USA, 415-662-2533

**Maurice Klapfish**, Venture Development Corp., One Apple Hill, Natick, MA, 01760-9904, USA, 508-653-9000

**Raymond Kolment**, Teknor Industrial Computers Inc., 616 Boivin, Boisbriand, J7G2A7, Canada, 514-437-5682

**Ori Kopelman**, Global Brain, Inc., 555 Bryant Street, #210, Palo Alto, CA, 94301, USA, 415-327-2012

**Richard Kupnicki**, Leitch Technology Centre, 21 Concourse Gate, Unit 16, Nepean, K2E 7S4, Canada, 416-445-9648



**David Lawrence**, SystemSoft, 200 South A St., Suite 208, Oxnard, CA, 93030, USA,  
805-486-6686

**Edwin Lee**, Pro-Log Inc., 23 Riesling Way, Scotts Valley, CA, 95066, USA, 408-461-1707

**Rick Lehtinen**, In-Stat, 7418 E. Helm Dr., Scottsdale, AZ, 85260, USA, 602-483-4472

**Steven Leibson**, EDN Magazine, 275 Washington St., Newton, MA, 02158, USA,  
617-558-4214

**Markus Levy**, EDN, 1936 Sheffield Dr, El Dorado Hills, CA, 95762, USA, 916-939-1642

**Adam Ley**, Texas Instruments, 6412 Hwy 75 S., M/S 838, Sherman, TX, 75090, USA,  
903-868-5761

**Jim Lipman**, EDN Magazine, 1447 Lennox Lane, Livermore, CA, 94550, USA, 510-606-1370

**Sean Long**, National Semiconductor, Mail Stop A1545, 2900 Semiconductor Dr.,  
PO Box 58090, Santa Clara, CA, 95052, USA, 408-721-3046

**Bob Lorentzen**, BottomLine Ventures, 2082 Fieldcrest Dr, Milpitas, CA, 95035, USA,  
408-262-7780

**Mike Maas**, Cypress Semiconductor, 2401 East 86th St., Bloomington, MN, 55425, USA,  
612-851-5060

**Duncan MacVicar**, MacVicar Associates, 1171 Buckingham Dr, Los Altos, CA, 94024, USA,  
415-962-8053

**Aamer Mahmood**, Cisco Systems, Bldg B-2, 170 W. Tasman Dr., San Jose, CA, 95134-1706,  
USA, 408-526-5278

**Tets Maniwa**, Integrated System Design Magazine, 5150 El Camino Real, Suite D31,  
Los Altos, CA, 94022, USA, 415-903-0503

**Kamal Mansharamani**, DCM Datasystems, Vikrant Tower, 4 Rajendra Place,  
New Delhi, 110 008, India, 91-11-5719967

**Trevor Marshall**, YARC Systems, C/O CHIARA RIVERA AT YARC,  
975 Business Center Circle, Newbury Park, CA, 91320, USA, 805-499-9444

**Bert McComas**, InQuest Market Research, 2162 E. Nantucket, Gilbert, AZ, 85234, USA,  
602-813-7785

**Don McCook**, Dolch Computer Systems, 3178 Laurelview Ct, Fremont, CA, 94538, USA,  
510-661-2220 x-204

**Rick McFarland**, SystemSoft, 2350 Mission College Blvd, Suite 450, Santa Clara, CA, 95054,  
USA, 408-988-6756 x24

**Michael McGowen**, Essential Communications, 4374 Alexander Blvd. NE, Suite T,  
Albuquerque, NM, 87107, USA, 505-344-0080

**John McGrath**, Intel Corp., Architecture Labs, 5200 NE Elam Young Pkwy, Hillsboro, OR,  
97124, USA

**Jonah McLeod**, Integrated System Design, 5150 El Camino Real, Suite D31, Los Altos, CA, 94022, USA, 415-903-0145

**Pierre McMaster**, Teknor Industrial Computers Inc., 616 Cure Boivin, Boisbriand, J7G 2A7, Canada, 514-437-5682

**Jim Medeiros**, Ziatech Corp., 1050 Southwood Dr., San Luis Obispo, CA, 93401, USA, 805-541-0488

**Dave Mendenhall**, Adaptec, 691 S. Milpitas Blvd, Milpitas, CA, 95035, USA, 408-957-6645

**W. Mentzer**, Intel Corp., PCI Components Division, 1900 Prairie City Rd., MS FM5-62, Folsom, CA, 95630, USA, 916-356-4705

**Scott Miller**, Dataquest, 251 River Oaks Pkwy, San Jose, CA, 95134, USA, 408-468-8460

**Frank Moldstad**, PC Graphics & Video, 201 E. Sandpointe Av., Suite 600, Santa Ana, CA, 92707, USA, 714-513-8437

**David Morgenstern**, Mac Week, 301 Howard St, 15th Floor, San Francisco, CA, 94105, USA, 415-243-3524

**Richard Mourn**, Symbios Logic, 1635 Aeroplaza Dr, Colorado Springs, CO, 80916, USA, 719-596-5795

**Harish Nayak**, PicoPower Technology—Cirrus Logic, 3100 W. Warren Ave., Fremont, CA, 94538, USA, 510-252-6292

**Jay Neer**, Molex Inc., Data/Com Division, 399 W. Camino Gardens Blvd, Suite 103, Boca Raton, FL, 33432, USA, 407-447-2907 x-3889

**Peter Nelson**, Toronto Microelectronics, Inc., 5149 Bradco Blvd, Mississauga, Ontario, L4W2A6, CANADA, 905-625-3203

**Paul Novell**, Cypress Semiconductor, 3901 North First St, San Jose, CA, 95134, USA, 408-943-2600

**Thomas Nygaard**, VMETRO, Inc., 16010 Barker's Point Ln. #575, Houston, TX, 77079, USA, 713-584-0728

**Richard O'Connor**, Tundra Semiconductor Corp., formerly NewBridge, 603 March Rd, Kanata, K2K 2M5, Canada, 613-592-0714

**Stephan Ohr**, Computer Design Magazine, 316 N. Chestnut St., Westfield, NJ, 07090, USA, 908-232-1380

**Daniel Palmans**, Hewlett-Packard, 5301 Stevens Creek Blvd, Santa Clara, CA, 95051, USA, 408-343-5134

**Jim Pappus**, USB Implementers Form, 2111 NE 25th Ave, Hillsboro, OR, 97124, USA,

**Phil Parker**, Number Nine Visual Technology, 18 Hartwell Ave., Lexington, MA, 02173, USA, 617-674-8513

**Tere' Parnell**, LAN Times, 441 E. Bay Blvd., Suite 100, Provo, UT, 84606, USA, 801-342-6801

**Joe Pavlat**, Pro-Log Corp., 12 Upper Ragsdale Drive, Monterey, CA, 93940, USA, 408-646-3511

**Jon Peddie**, PC Graphics Report, 4 St. Gabrielle Court, Tiburon, CA, 94920, USA, 415-435-1775

**Dave Podsiadlo**, AMCC, 6195 Lusk Blvd, San Diego, CA, 95121, USA, 619-535-4279

**J. Gerry Purdy**, Mobile Computing Insights, Inc., 20863 Stevens Creek Blvd, Suite 320, Cupertino, CA, 95014, USA, 408-777-4852

**Said Rahmani-Khezri**, Pathlight Technology, 767 Warren Road, Ithaca, NY, 14850, USA, 607-266-4000

**K.K. Rao**, Mylex, 34551 Ardenwood Blvd, Fremont, CA, 94555, USA, 510-796-6050 x-215

**Selliah Rathnam**, Philips Semiconductors, Trimedia product group, 811 E Arques Ave, Sunnyvale, CA, 94088, USA, 408-991-2868

**Dave Reed**, Nvidia Corp., 1228 Tiros Way, Sunnyvale, CA, 94086, USA, 408-720-6100

**Martin Reynolds**, Dataquest, 251 River Oaks Pkwy, San Jose, CA, 95134, USA, 408-468-8212

**Desi Rhoden**, VLSI Technology, 8375 South River Parkway, Tempe, AZ, 85284, USA, 602-752-3323

**Tracy Richardson**, Digital Semiconductor, Digital Equipment Corp., 77 Reed Rd., Hudson, MA, 01749, USA, 508-568-5103

**Dave Ridgeway**, Xilinx Inc., 2100 Logic Dr, San Jose, CA, 95124, USA, 408-559-7778

**Dwight Riley**, Compaq Computer Corp., MS 100505, PO Box 692000, Houston, TX, 77269, USA, 713-518-4367

**Jack Roberts**, Dataquest Inc., 251 River Oaks Pkwy, San Jose, CA, 95134, USA, 408-468-8539

**Kim Rubin**, GreenSpring Computers, 1204 O'Brien Drive, Menlo Park, CA, 94025, USA, 415-327-1200

**Scott Ruple**, Emulex, 3535 Harbor Blvd, Costa Mesa, CA, 92626, USA, 800-EMULEX3

**Arthur Ryan**, National Instruments, 6504 Bridge Point Pkwy, Austin, TX, 78730, USA, 512-433-8845

**Kelly Ryer**, MacWeek, 301 Howard St., 15th Floor, San Francisco, CA, 94105, USA, 415-243-8524

**Mike Salameh**, PLX Technology, 625 Clyde Ave, Mountain View, CA, 94043, USA, 415-960-0448

**Bill Samaras**, Intel Corp., 1606 Knollwood Ave, San Jose, CA, 95125, USA, 408-765-0824

**Ron Sartore**, 11511 Eastridge Pl., San Diego, CA, 92131, USA, 619-549-1290

**Sanjay Sawant**, Quickturn Design Systems, 440 Clyde Ave, Mountain View, CA, 94043, USA, 415-967-3300

**Frank Schapfel**, Digital Semiconductor, Digital Equipment Corp., 77 Reed Rd., HLO2-1/H12, Hudson, MA, 01749, USA, 508-568-4861

**Harold Schiefer**, Genesis Microchip Inc., 719-35 Ormskirk Ave, Toronto, 1A8M6S, CANADA, 905-470-2742

**Walter Scott**, IBM Corp., 11400 Burnet Road, Austin, TX, 78758, USA, 512-838-7268

**Fernando Serra**, Imaging Technology, Inc., 55 Middlesex Turnpike, Bedford, MA, 01730, USA, 617-275-2700 X330

**Joseph Sgro**, Alacron, 71 Spitbrook Rd, Suite 204, Nashua, NH, 03060, USA, 603-891-2750

**Tony Shaberman**, Intel Corp., FlashCard System Group, 1900 Prairie City Rd, FM3-77, Folsom, CA, 95661, USA, 916-356-7399

**Rob Shaddock**, Loughborough Sound Images, Loughborough Park, Ashby Rd, Loughborough, Leicestershire, LE11 3ne, England, 011-44-1509634365

**Tom Shanley**, MindShare, Inc., 2202 Buttercup Dr, Richardson, TX, 75082, USA, 214-231-2216

**Gerritt Slavenberg**, Philips Semiconductor, Trimedia, 811 E. Arques Ave, PO Box 3409, Sunnyvale, CA, 94088, USA, 408-991-4974

**Brian Small**, QuickLogic, 2933 Bunker Hill Lane, Santa Clara, CA, 95054, USA, 408-987-2003

**Chuck Small**, Hewlett-Packard, 1900 Garden of the Gods Rd, Colorado Springs, CO, 80901, USA, 719-590-2006

**Niel Smith**, Pacific Technology Group, 4701 Patrick Henry Dr., Suite 2101, Santa Clara, CA, 95054, USA, 408-764-0644

**Dennis Snyder**, Symbios Logic, 1630 Aeroplaza Dr, Colorado Springs, CO, 80916, USA, 719-573-3573

**Cary Snyder**, Virtual Chips, 2107 N. First St., Suite 100, San Jose, CA, 95131, USA, 408-452-1600x214

**Jeff Solliday**, Solliday Engineering Corp., 1756 18th St, San Francisco, CA, 94107, USA, 415-621-0616

**Margit Stearns**, Symbios Logic, 1635 Aeroplaza Dr, Colorado Springs, CO, 80916, USA, 719-573-3228

**Roger Storer**, Second Wave, Inc., 2525 Wallingwood Dr, Bldg. 13, Austin, TX, 78746, USA, 512-329-9283

**Clyde Thomas**, Allen-Bradley, Automation Group, 1 Allen-Bradley Drive, Mayfield Heights, OH, 44124, USA, 216-646-4402

**Tom Thorsteinson**, Linear Systems, 959 Powell Ave, Winnipeg, R3H 0H4, Canada, 204-632-4300

**Michele Tidwell**, IBM, Storage Systems Division, 5600 Cottle Rd, San Jose, CA, 95193, USA, 408-256-3874

**Roger Tipley**, Compaq Computer Corp., PO Box 692000, Mail Stop 050708, Houston, TX, 77269, USA, 713-374-6691

**Neil Trevett**, 3Dlabs, Inc., 181 Metro Drive, Suite 520, San Jose, CA, 95110, USA, 408-436-3456

**Dawn Tse**, Mylex Corp., 34551 Ardenwood Blvd, Fremont, CA, 94555, USA, 510-796-6050

**Mike Turay**, Multimedia PC Product Group, Philips Semiconductors, 800 E. Middlefield Rd., Mountain View, CA, 94043, USA, 415-335-2546

**Rajesh Vashist**, Adaptec Inc., 691 S. Milpitas Bl, Milpitas, CA, 95035, USA, 408-957-4869

**Giri Venkat**, Yamaha Systems Division, 100 Century Center Court, San Jose, CA, 95112, USA, 408-437-3133

**Dick Vignoni**, Digital Equipment Corp., 77 Reed Rd., Hudson, MA, 01749-2895, USA, 508-568-5179

**Deborah Vogt**, Digital Semiconductor, Digital Equipment Corp., 77 Reed Rd., Hudson, MA, 01749, USA, 508-568-6328

**Eyal Waldman**, Galilio Technology, 1735 N. First St. #308, San Jose, CA, 95112, USA, 408-451-1400

**Hans Weersch**, Philips Semiconductor, 811 E. Arques Ave., MS 52, PO Box 3409, Sunnyvale, CA, 94088, USA, 408-991-4507

**Ray Weiss**, Computer Design, 22830 Ostronic Dr, Woodland Hills, CA, 91367, USA, 818-704-9454

**George White**, Corollary, Inc., 2802 Kelvin Avenue, Irvine, CA, 92714, USA, 714-250-4040

**David Wilner**, Wind River Systems, 5000 W Chandler Blvd, Chandler, AZ, 85226, USA,

**Tom Wilson**, Tundra Semiconductor Corp., formerly Newbridge Microsystems, 603 March Rd., Kanata, K2K2M5, CANADA, 613-592-0714

**Ron Wilson**, EE Times, 2800 Campus Dr, San Mateo, CA, 94403, USA, 415-525-4498

**Diana Wilson**, Intel Corp., Mail Stop FM4-18, 1900 Prairie City Rd., Folsom, CA, 95630, USA

**Lee Wilson**, IBM, Power Personal Systems, 11400 Burnet Rd, Mail Stop 4357, Austin, TX, 78758, USA, 512-838-6569

**Mark Wodyka**, AITech International, 47971 Fremont Blvd., Fremont, CA, 94538, USA, 510-226-8960, x113

**Leo Wong**, Altera Corp, 3 W. Plumeria, San Jose, CA, 95134, USA, 408-894-7893

**Shawn Worsell**, V3 Corp., 2348 Walsh Ave., Suite G, Santa Clara, CA, 95051, USA, 408-988-1050

**Maury Wright**, EDN, 12544 Robison, Poway, CA, 92064, USA, 619-748-6785

**Yong Yao**, Micro Design Resources, 480 San Antonio Rd, Suite 210, Mountain View, CA, 94040, USA, 415-917-3066

**Clayton Yee**, Philips Semiconductor, Interactive TV Group, 811 E. Arques Ave, PO Box 3409, Sunnyvale, CA, 94088, USA, 408-991-2986





0-929392-34-5