



# SIGDA NEWSLETTER

SPECIAL INTEREST GROUP ON DESIGN AUTOMATION

Volume 6

Number 2

June 1976

## Contents:

Chairman's Message	1
From the Editor	1
Meeting Announcements and Calls for Papers	2
Recent Publications - W. M. vanCleemput	12
A Model of Real Time Control System Production - M. N. Matelan	14
Computer Aided Design and Design Automation in Europe - Part 2 Waldo G. Magnuson, Jr.	62

# SIGDA

## ACM Special Interest Group on Design Automation

### ADDRESSES

#### CHAIRMAN:

Charles W. Rose  
Computing & Information Science  
Case Western Reserve University  
Cleveland, Ohio 44106  
(216) 368-2800

#### VICE-CHAIRMAN:

Judith G. Brinsfield  
Bell Laboratories  
Building 3B-323  
Whippany, NY 07981  
(201) 386-3169

#### SECRETARY-TREASURER:

Luther C. Abel  
Digital Equipment Corporation  
146 Main Street  
Maynard, Mass. 01754  
(617) 897-5111

#### EDITOR:

Robert J. Smith, II  
Lawrence Livermore Laboratory, L-156  
P.O. Box 808  
Livermore, CA 94550  
(415) 447-1100, X-8088

#### TECHNICAL COMMITTEE:

Judith G. Brinsfield

#### MEMBERSHIP COMMITTEE:

William M. vanCleave  
Digital Systems Laboratory  
Stanford University  
Stanford, CA 94305  
(415) 497-1270

#### PUBLICITY COMMITTEE:

Lorna Capodanno, Chairman  
Bell Labs  
2C169  
Murray Hill, New Jersey 07974  
(201) 582-6909

#### BOARD OF DIRECTORS:

Dr. John Grason  
Department of Electrical Engineering  
Carnegie Mellon University  
Pittsburgh, Pennsylvania 15213

Dr. Edwin Hassler  
Design Automation Department  
Components Group  
Texas Instruments, Inc.  
P.O. Box 5012  
Dallas, Texas 75222

Professor James Linders  
Department of Applied Analysis  
University of Waterloo  
Waterloo, Ontario  
Canada

#### BOARD OF DIRECTORS, cont.'d

Dr. Ralph Miller  
Structures Research  
Commercial Airplane Group  
Boeing Company  
P.O. Box 3707  
Seattle, Washington 98124

Dr. Jean-Marie Comeau  
Bureau of Management Consulting  
365, Ave Laurier Ouest  
Ottawa, Canada  
K1A0T5

### MEMBERSHIP

SIGDA dues are \$3.00 for ACM members and \$5.00 for non-ACM members. Checks should be made payable to the ACM and may be mailed to the SIGDA Secretary/Treasurer listed at left, or to SIGDA, ACM Headquarters, 1133 Avenue of Americas, New York, N.Y. 10036. Please enclose your preferred mailing address and ACM Number (if ACM member).

### SIG/SIC ACTIVITIES

- 1) Informal technical meetings at NCC.
- 2) Formal meeting during National ACM meeting + DA Workshop.
- 3) Joint sponsorship of annual Design Automation Workshop.
- 4) Quarterly newsletter.
- 5) Panel and/or technical sessions at other National meetings.

### FIELD OF INTEREST OF SIGDA MEMBERS

Theoretic, analytic, and heuristic methods for:

- 1) performing design tasks,
- 2) assisting in design tasks,
- 3) optimizing designs through the use of computer techniques, algorithms and programs to:
- 1) facilitate communications between designers and design tasks,
- 2) provide design documentation,
- 3) evaluate design through simulation,
- 4) control manufacturing processes.

### CHAIRMAN'S MESSAGE

In January, after much work by your treasurer, Luther Abel, SIGDA submitted its budget for the 1976-1977 fiscal year. During this process, it became clear that a recent decision by the ACM council to increase the headquarters assessment on non-ACM SIG members (class (2)) will cut into our usual operating surplus. The new assessment is \$7.00 per head while our dues for these members, of which there are now 37, is \$5.00.

SIGDA is fiscally healthy, deriving income from sponsorship of workshops and conferences, especially the June Design Automation Conference as well as from dues. This increased assessment is not going to put us in the red, but it does seem logical that non-ACM SIG members should "pay their own way" in terms of assessments and Newsletter mailings in the same manner that ACM SIG members do.

Changing the dues structure for SIGDA is no easy task, however. The bylaws of SIGDA place an upper bound on the dues as stated in Article III C.

"Each member shall pay dues as determined by the executive committee of the Group. These dues shall not exceed \$3.00 for members in class (1) or \$5.00 per calendar year for members in class (2)."

Increasing the dues for non-ACM members beyond \$5.00 requires a change in the bylaws as specified in Article VIII.

- a. "An amendment to these bylaws may be proposed by a resolution adopted by the Executive Committee or by a petition signed by at least 10 members.
- b. Within 60 days after receipt by the Secretary-Treasurer of the proposal of an amendment, the Secretary-Treasurer shall conduct a mail vote on the amendment. The proposed amendment shall be reviewed by the Chairman of the Committee on Special Interest Groups and Committees prior to the mail vote on the amendment. Ballots shall be mailed to all members of the Group and shall state the last day for receipt of a voted ballot by the Secretary-Treasurer. This data shall be 30 to 60 days after the last ballots are mailed. The amendment shall become effective if it receives a majority vote of one-fourth of the Group membership, with the effective date falling 60 days after the final receipt date stated on the amendment ballots."

It is my opinion that to set the dues in the bylaws is not wise since SIGDA is, in effect, a slave to headquarters assessments, and must respond in a fairly short time, while the bylaws amending process is involved, expensive and lengthy. It is already too late to increase non-ACM member dues for 1976 renewals.

On the advice of your Executive Committee and Board of Directors, I will place before you an amendment to the bylaws replacing article IIIC with one allowing dues to be set by the Executive Committee with the advice of the Board of Directors with no stated limit.

The intent is not to give your officers the license to levy unreasonable dues, but to allow us to respond in a timely manner to Headquarters. Since the inception of SIGDA, the dues have covered only the expenses associated with each member plus office expenses such as mailing lists, printing of brochures, etc. Our surplus has accrued from conference proceeds and is in no way "hard money" which we can count on from year to year. The vote will be held during the summer in accordance with the procedures set down by ACM and our bylaws. We'll discuss the situation more fully at the DA Conference in June.

I am happy to report that I have been contacted by Dr. Roy Russo of IBM Research, the recently appointed head of the IEEE Technical Committee on Design Automation, who is very interested in joint efforts between our organizations. We are currently exploring the possibility of a joint workshop on design aids for microprocessors and microprocessor-based systems to be held during the week of COMPCON in February 1977 at San Francisco. The Newsletter will keep you posted on developments.

On a less happy note, I was informed recently that the sessions which SIGDA proposed for ACM '76 have been cancelled for lack of papers. Apparently only 2 papers were submitted in response to personal invitations and the general call for papers. While I am pleased at SIGDA's response to the DA Conference and Workshops, we are a part of ACM, and to not participate in the national conference is an embarrassment. I hope that we can generate enough interest for a session at ACM '77, perhaps relating to computer sciences aspects of Design Automation.

Hope to see all of you at the DA Conference in San Francisco.

### FROM THE EDITOR

The bulk of the material in this issue was contributed by two of my colleagues at LLL: thanks to Nick Matelan and Waldo Magnuson for two interesting articles.

Once more, I'd like to renew my plea to those of you with manuscript material of interest to the

SIGDA membership: send me Newsletter material! We at LLL continue to supply a disproportionate share of the Newsletter copy -- primarily because of a lack of other inputs. Will you contribute?

*Rob Smith*  
5-10-76

WORKSHOP ON DESIGN AUTOMATION AND MICROPROCESSORS

CALL FOR PARTICIPATION

The IEEE Technical Committee on Design Automation and the ACM Special Interest Group on Design Automation plan to organize a workshop on Design Aids for the Design of Microprocessors and Microprocessor-based Systems.

This 2 or 3-day workshop is tentatively scheduled for February 1977 on the West Coast. The intention is to hold the workshop as close as possible to the COMPCON 77 Spring Meeting.

The workshop will have a formal part with refereed contributions and invited talks. These will be published in the proceedings of the Workshop. There will be a number of informal sessions. In order to stimulate open discussion these will not be part of the proceedings and no taperecorders or cameras will be allowed during these sessions.

Since this is intended to be a workshop (or a working conference) the number of attendants will be strictly limited. If you intend to participate by submitting a paper, by talking at an informal session or just by attending, please indicate this by sending the coupon below to the General Chairman.

Some Important Deadlines: Refereed Papers due: October 1, 1976; Notice of acceptance sent: November 15, 1976; Final manuscripts due: December 31, 1976; Participation in informal sessions: December 15, 1976.

---

WORKSHOP ON DESIGN AUTOMATION AND MICROPROCESSORS  
FEBRUARY 1977, SAN FRANCISCO (TENTATIVE)

NAME \_\_\_\_\_

COMPANY \_\_\_\_\_

DEPARTMENT \_\_\_\_\_

STREET ADDRESS \_\_\_\_\_

CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP \_\_\_\_\_

COUNTRY \_\_\_\_\_

Please check one or more boxes

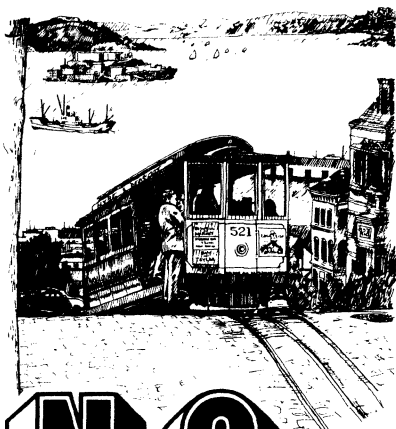
I plan to submit a paper. Tentative Title: \_\_\_\_\_

I am willing to participate in an Informal Session on \_\_\_\_\_

I am planning to attend this workshop

None of the above, but place me on your mailing list.

Send to: W. M. vanCleemput, Digital Systems Laboratory, Stanford University,  
Stanford, California 94305



# NO. 13 Design Automation Conference

**Jack Tar Hotel**  
Van Ness at Geary  
San Francisco,  
California 94101

## Registration Fee

	Advance Registration	Registration at Conference
ACM, IEEE Members	\$50	\$60
Nonmembers	\$65	\$75
Students*	N/A	\$20

This fee includes two luncheons, coffee service and one copy of the Conference Proceedings. Early registration is recommended. All advance registration forms must be received no later than June 4, 1976. After that date, registrations will be accepted at the Conference registration desk.

\*Full time students with card identification can register only at the Conference for \$20. The fee includes admission to the sessions and one copy of the Conference Proceedings. Luncheon tickets may be purchased separately.

## Advance Registration

Advance registration closes June 4, 1976. After that date, registrations will be accepted at the DAC Conference registration desk. Your check must accompany this form. Kindly mail the form with a check made payable to "1976 DA Conference" to the Registration Chairman:

P. O. Pistilli  
Bell Laboratories  
11900 N. Pecos St.  
Denver, Colorado 80234  
Tel.: 303 451-4364

Name (last name first) \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City/State \_\_\_\_\_

Zip \_\_\_\_\_

Use this form or a facsimile for advance registration. Enclosed is a check for advance registration (fee includes two luncheons, coffee service and one copy of the Conference Proceedings).

Member ACM or IEEE \_\_\_\_\_ \$50  
Nonmember \_\_\_\_\_ \$65  
ACM/IEEE Membership Number \_\_\_\_\_

	Monday	Tuesday	Wednesday
8:00	Interactive Graphics Tutorial	Interconnection Tutorial	Software Engineering Tutorial
9:00	Introductory Remarks and Keynote Address	SESSION 5 Modeling and CAD	SESSION 6 Architectural DA
10:30	Coffee	SESSION 7 Interconnection	SESSION 12 Design Rule Verification
11:00	Program Overview	SESSION 8 Testing and Testers	SESSION 13 Software Engineering I
12:00	Lunch	Luncheon with Speaker Dave Snyder Walt Disney Productions	SESSION 14 Computer Aided Documentation
1:30	SESSION 1 Structural and Mechanical DA	SESSION 9 Placement and Routing	SESSION 15 IC Layout
3:00	Coffee	SESSION 10 Fault Tolerant DA	SESSION 16 Software Engineering II
3:30	SESSION 2 Digital Logic Simulation	SESSION 11 Data Bases	SESSION 17 Software Reliability
5:00	No Host Cocktail Party	No Host Cocktail Party	No Host Cocktail Party
6:30	Common Interest Groups	Common Interest Groups	Common Interest Groups

## Hotel Registration

Thirteenth Design Automation Conference  
June 28-30, 1976

Please mail this form or a facsimile.

All reservations will be confirmed. Please insure that your reservations are received prior to June 11 so as to take advantage of the pre-established rates.

Circle Daily Rate Desired

Single \$28  
Double \$30  
Twin \$34

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City/State \_\_\_\_\_ Zip \_\_\_\_\_

Arrival Date \_\_\_\_\_ Time \_\_\_\_\_ A.M. P.M.

Departure Date \_\_\_\_\_ Time \_\_\_\_\_ A.M. P.M.

Send To:  
Jack Tar Hotel  
Van Ness at Geary  
San Francisco, California 94101  
Attention: Reservation Manager

Toll Free Phone:  
800-227-4730  
Always mention DAC '76.

## Proceedings

The proceedings will contain seventy-one papers and all significant visual material for each paper. Each registrant will receive one copy of the Proceedings at the Conference prior to the start of sessions. Additional copies may be purchased at the Conference (on a cash and carry basis) for \$10.00. After the Conference, mail orders are to be sent to either of the following organizations. These organizations should be contacted before placing your order to determine cost and availability of the Proceedings.

ACM  
Order Department  
1133 Avenue of the Americas  
New York, N.Y. 10019  
IEEE Computer Society  
ATTN: Mr. True Seaborn  
5855 Naples Plaza, Suite 301  
Long Beach, California 90803

Copies of the Proceedings of past Conferences, when available, can be obtained from either of the above organizations at their prevailing price list.



## Acknowledgement

The 13th Design Automation Conference is sponsored by the ACM (Association for Computing Machinery) Special Interest Group on Design Automation and the IEEE (Institute of Electrical and Electronics Engineers) Computer Society Design Automation Technical Committee.

## Registration Hours

All persons attending the Conference will be required to register. The registration desk will be open during the following hours at the Jack Tar Hotel.

<b>Sunday June 27</b>	<b>7:00 p.m.-9:00 p.m.</b>
<b>Monday June 28</b>	<b>7:30 a.m.-5:00 p.m.</b>
<b>Tuesday June 29</b>	<b>8:00 a.m.-3:00 p.m.</b>
<b>Wednesday June 30</b>	<b>8:00 a.m.-10:00 a.m.</b>

## Headquarters Hotel

All sessions of the DAC will be held at the Jack Tar Hotel, Van Ness at Geary, San Francisco, California 94101. It is recommended that you come into the San Francisco International Airport and take a Downtown Airline Terminal bus which costs about \$1.25 and leaves every 20 minutes. The downtown terminal is about eight blocks from the Jack Tar Hotel, and thus an inexpensive taxi trip away.

## Housing

The DAC Committee will not make room reservations. A block of rooms has been reserved at the Jack Tar Hotel. Kindly communicate directly with the hotel. A hotel reservation form is provided in this program for your convenience. When using company or private stationary, please be sure to specify that you will be attending the 13th Design Automation Conference. In making your plans, please notice that this year we have 3 FULL DAYS of technical program.

## Dress

Lightweight wools and knits are right for June in San Francisco. The thermometer seldom goes above 65° and the low is usually around 58°. But during the summer months temperatures rise from 20° to 30° within a small distance of the city, so if you are planning a trip down the peninsula or across the Golden Gate Bridge to the nearby wine country, bring along some summer cottons.

## About The Conference

Design Automation implies the use of computers as tools which aid the design process. It is often extended to include areas such as testing, simulation and certain portions of manufacturing. These are generally referred to as computer aided design, computer aided manufacturing, machine aids, automated design, automated drafting, et cetera and are parts of potentially totally integrated systems.

It is the purpose of this conference to provide a forum whereby developments in the field of Design Automation can be presented and discussed. It is the aim of this conference to improve and expand the quality and quantity of Design Automation developments. Because of the breadth of subject matter, the three day program has been so arranged that the technical papers are presented concurrently in different disciplines of Design Automation. For some there may be conflicts, but for most there will be something of interest throughout each day.


Rooms are available for Common Interest Groups ("birds of a feather") to convene after each day of formal presentations. Discussions may involve such topics as circuit design, packaging, interconnection, testing and simulation, architecture, mechanical, software engineering, or whatever topic is of interest to a group. Individuals desiring a particular Common Interest Group topic may have a session scheduled and notice posted by making the request at the registration desk. It is here where innovation can take place. However, it is up to the attendees to make the Common Interest Group meetings successful. It is anticipated that everyone attending the conference will leave having gained something.

## Sightseeing

San Francisco is one of the world's most visited and talked about cities. Practically every block is interesting in one way or another and when the sun breaks through the morning fog, the cityscape becomes a pattern of Mediterranean pastels unlike any other city in North America. It's cosmopolitan living at its best with fine hotels and restaurants, museums, excellent opera, theatre and music, fascinating shops, exciting night life — everything you expect of a great city. So we suggest that you — ride the cable cars . . . explore Chinatown . . . see colorful Fisherman's Wharf . . . visit Mission Dolores, the Maritime Museum and the Balclutha . . . shop in Ghirardelli Square and Maiden Lane . . . walk in Union Square . . . or just climb to the top of the city and look out and down upon the breathtaking panorama all around you. Also, adjacent to San Francisco, a new large amusement park has opened as a Disneyland-like celebration of America, called Great America.


## Committee

### General Chairman




Donald J. Humcke  
Bell Laboratories  
Bldg. 1F-214  
Holmdel, New Jersey 07733  
201-949-6253

### Vice Chairman




J. Michael Galey  
IBM Corporation  
Dept. R06, Bldg. 061A  
Monterey and Cottle Roads  
San Jose, California 95193  
408-256-6606

### Program Chairman




Stephen A. Szygenda  
The University of Texas  
Electrical Engineering Department (ENS515)  
Austin, Texas 78712  
512-471-7365

### Finance Chairman




P.O. Pistilli  
Bell Laboratories  
Building 1F-66  
11900 N. Pecos St.  
Denver, Colorado 80234  
303-451-4364

### Publicity Chairman




Nitta P. Dooner  
IBM T. J. Watson Research Center  
P.O. Box 218  
Yorktown Heights, New York 10598  
914-945-1097

### Publications Chairman




Judith G. Brinsfield  
Bell Laboratories  
Building 3B-323  
Whippany Rd.  
Whippany, New Jersey 07981  
201-386-3169

### Arrangements Chairman




J. S. Olila  
Signetics  
811 E. Arques Ave.  
Sunnyvale, California 94086  
408-739-7700

### IEEE Representative



Stephen Pardee  
Bell Laboratories  
Building 3B-310  
Whippany, New Jersey 07981  
201-386-5840

### ACM Representative

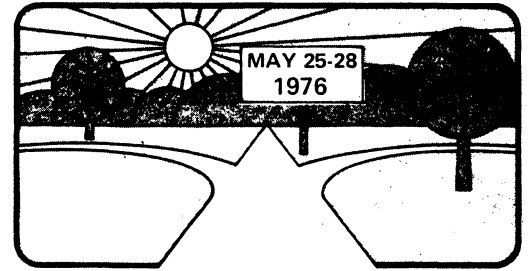


David W. Hightower  
Bell Laboratories  
Building 2B-312A  
Holmdel, New Jersey 07733  
201-949-6549

# THE SIXTH INTERNATIONAL SYMPOSIUM ON MULTIPLE-VALUED LOGIC

May 25-28, 1976

University Center • Utah State University • Logan, Utah



25 YEARS OF SERVICE

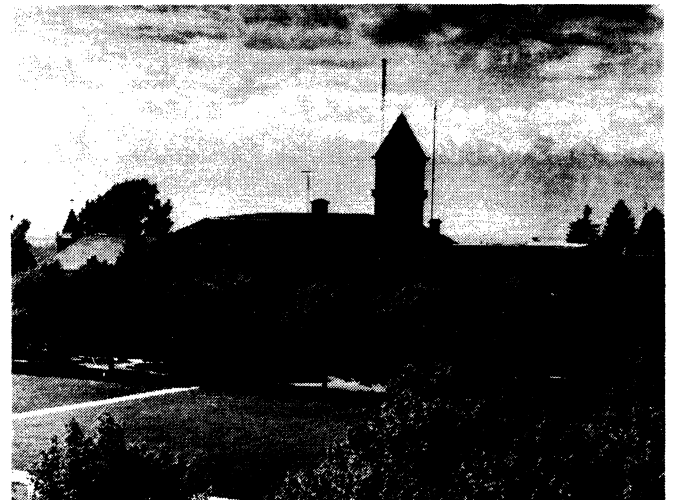


IEEE COMPUTER SOCIETY

acm



ONR



Cosponsored by: ACM, IEEE Computer Society, Office of Naval Research, and Utah State University

For further information, contact Stephen Su, Electrical Engineering Dept., U.S.U., Logan, Utah 84322. Telephone (801) 752-4100, ext. 7806 (office); (801) 752-6717 (home)

Tuesday, May 25      Baugh Motel, Conference Room  
6:30-8:30 p.m.      Registration and get-acquainted hour  
8:30-9:30 p.m.      Planning Session for future symposium  
Wednesday, May 26      Tutorial Sessions, Walnut Room, University Center  
8:30-9:00 a.m.      *Introduction:* S. Y. H. Su, Symposium Chairman, Z. Vranesic, Program Chairman  
*Welcome:* G. L. Taggart, President of U.S.U., E. J. Middlebrooks, Dean of Engineering, K. Baker, Head of Electrical Engineering  
9:00-9:50 a.m.      "A Survey of Multiple-valued Algorithmic Logics: From a Practical Point of View," D. Rine, West Virginia U.  
10:10-11:00      "On Algebraic and Combinatorial Foundations of Multiple-valued Logics," I. G. Rosenberg, U. of Montreal  
11:10-12:00      "Cubical Representation for Computer-aided Synthesis of Multiple-valued Switching Functions," S. Y. H. Su, U.S.U., and P. T. Cheung, Packard Instruments  
1:30-2:20 p.m.      "Circuits for Multiple-valued Logic—A Tutorial and Appreciation," K. C. Smith, U. of Toronto  
2:50-3:40      "Multiple-valued Charge-Coupled Devices," U. Strasilla, Swiss Federal Institute of Technology

3:40-5:00      "Applications of Multiple-valued Logic," (panel)  
Chairman: M. S. Michalski  
Panelists: B. R. Gaines, S. Haack, T. Kitahashi, W. J. Poppelbaum, D. Rine, and K. C. Smith  
5:00-6:00      Social Hour  
Thursday, May 27      (Session A: Walnut Room; Session B: Room 329)  
8:30-10:30 a.m.      Session 1  
(1A) "Multi-valued Multi-threshold Networks," O. Ishizuka, Miyazaki U.  
"Cellular and Hybrid Threshold Networks," C. Moraga, U. of Dortmund  
"Three Cell Structures for Ternary Cellular Arrays," J. A. Bate and J. C. Muzio, U. of Manitoba  
"The Implementation of a Multi-valued Parallel Divider Array," J. Newton, U. of Toronto  
(1B) "A Multi-valued Switching Algebra with Boolean Properties," J. Dussault and G. Metzger, U. of Illinois-Urbana; M. Krieger, U. of Ottawa  
"Identification of Different Functional Properties of Multiple-valued Switching Functions," P. T. Cheung, Packard Instrument Co., Inc.  
"Logic Properties of Unate and of Symmetric Discrete Functions," J. P. Deschamps and A. Thayse, MBLE Research Laboratory  
"Further Results on m-TMC Forms for Multi-valued Function," K. L. Kodandapani, and D. K. Pradhan, U. of Regina  
11:00-12:00      Session 2—Invited Talk, Walnut Room  
"Associative Memories as Means for Implementing Multiple-valued Logic," Y. H. Pao and J. Altman, Case Western Reserve U.

- 1:00-2:00 Tour of U.S.U. Facilities  
 2:00-3:00 Session 3
- (3A) "Single Fault Detection in Multi-valued Combinational Circuits," R. Spillman, Utah State U.  
 "Synthesis of P-valued Asynchronous Sequential Circuits by Using a General Clock Function," J. L. Huertas and J. K. Acha, U. of Sevilla
- (3B) "Every Reduct of Two-element Boolean Algebra can be Finitely Axiomatized by Modus Ponens or Substitution Rule," S. J. Surma.  
 "Classical Indeterminacy, Many-valued Logic, and Super Valuations," J. N. Martin, U. of Cincinnati
- 3:30-5:00 Session 4
- (4A) "A Study on the Implementation of Three-valued Logic," H. T. Mouftah, U. of Toronto  
 "Static-hazard-free T-gate for Ternary Memory Element and Its Application to Ternary Counters," M. Kameyama and T. Higuchi, Tohoku U.  
 "Ternary Logic in a Positional Control System," H. T. Mouftah and K. C. Smith, U. of Toronto
- (4B) "Decisive Venn Diagrams," G. Epstein, Indiana U.  
 "Many-valued Propositional Intuitionism," C. G. Morgan, U. of Victoria  
 "A Multi-step Formation of Variable Valued Logic Hypotheses," J. Larsen, U. of Illinois-Urbana
- 5:00-7:00 Social Hours  
 7:00-10:00 Banquet
- Friday, May 28
- 8:30-10:30 a.m. Session 5
- (5A) "Two-place Decomposition and the Synthesis of Many-valued Switching Circuits," D. M. Miller, U. of New Brunswick; J. C. Muzio, U. of Manitoba  
 "Noise Margins in Multi-valued Logic Circuits," D. Etemble, U. of Paris VI; M. Israel, CNAM Paris

- "On Multi-valued Complete Codes," T. Kitahashi and K. Tanaka, Osaka U.; S. Kaneyori, Nomura Coordinate Research Centre  
 "The Relationship Between Computational Circuit Complexity and Radix," J. R. Armstrong, Virginia Polytechnic Institute and State U.
- (5B) "Fuzzy Reasoning and the Logic of Uncertainty," B. R. Gaines, U. of Essex  
 "Fuzzy Maps and Their Applications in the Simplification of Fuzzy Switching Functions," A. Kandel, New Mexico Institute of Mining and Technology  
 "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," E. H. Mamdani, U. of London  
 "N-variable Fuzzy Maps with Application to Disjunctive Decomposition of Fuzzy Switching Functions," G. Schwede, New Mexico Institute of Mining and Technology
- 11:00-12:00 Session 6—Invited Talk, Walnut Room  
 "Semantic Influence from Fuzzy Premises," L. A. Zadeh, U. of Calif. Berkeley
- 1:30-3:00 Session 7
- (7A) "The Binary Representation of m-valued Logic with Applications to Universal Decision Elements," J. Loader, Brighton Polytechnic  
 "Application of Multi-valued Logics to the Study of Human Movement Control and of Movement Disorders," L. Kohout, U. of London  
 "An Application of Variable-valued Logic to Inductive Learning of Plant Disease Diagnostic Rules," R. Chilausky, B. Jacobsen, R. S. Michalski, U. of Illinois-Urbana
- (7B) "A Minimization Algorithm for Ternary Switching Functions," T. C. Yang and A. Wojcik, Illinois Institute of Technology  
 "Multiple-output Multi-valued Prime Implicants," W. R. Smith and H. Shah, U. of Bridgeport  
 "Generation of Value-consistent Multi-valued Prime Implicants," W. R. Smith and R. W. Duggan, U. of Bridgeport
- 3:30-5:10 Session 8—Recent Results

## GUIDED TOURS

- Yellowstone and Grand Teton Parks—\$375/person double occupancy, 6 days/5 nights: includes meals, lodging and transportation, scenicrider ride on Lake Yellowstone, float trip on Snake River, tram ride in Teton Village and a melodrama at Jackson Hole.
- Bryce, Zion and Grand Canyon National Parks and Las Vegas—\$261/person double occupancy, 5 days/4 nights: includes meals, lodging and transportation, scenic tours of the parks and one night club and dinner show in Las Vegas.

For information, contact Colburn Travel Service, 60 E. Center, Logan, Utah 84321 (801) 752-6115

## LOCAL ACCOMMODATIONS

Reserve your room with the motels before May 20th to guarantee a place to stay. Rooms will be held until 6:00 p.m. on day of arrival unless deposit of the first night's rental is made.

Baugh Motel		Holiday House	
1 person	\$13	1 person	\$13
2 persons	\$18	2 persons	\$16

Baugh Motel—153 S. Main, Logan, Ut. 84321 (801) 752-5220

Holiday House—447 N. Main, Logan, Ut. 84321 (801) 752-9141

## LOCAL TRANSPORTATION

A U.S.U. bus will provide free transportation from the Salt Lake Airport to Logan

May 25	lv. SLC 3 p.m.	ar. Logan 5 p.m.
May 25	lv. SLC 6 p.m.	ar. Logan 8 p.m.
May 29	lv. Logan 10 a.m.	ar. SLC 12 noon

Plane service from Key Airlines:

SLC to Logan	8:00 a.m.—8:50 a.m. 4:45 p.m.—5:10 p.m.	Logan to SLC	10:01 a.m.—10:25 a.m. 6:15 p.m.—7:15 p.m.
--------------	--	--------------	--

Greyhound bus leaves SLC at 8:15 a.m., 3:40 p.m., 8:15 p.m.

SESSIONS	IEEE or ACM MEMBERS		NON-MEMBER	
	Before May 5	After May 5	Before May 5	After May 5
(1) Tutorial	<input type="checkbox"/> \$20	<input type="checkbox"/> \$25	<input type="checkbox"/> \$25	<input type="checkbox"/> \$30
(2) Tech Session	<input type="checkbox"/> \$40	<input type="checkbox"/> \$50	<input type="checkbox"/> \$50	<input type="checkbox"/> \$60
(3) Both (1) & (2)	<input type="checkbox"/> \$50	<input type="checkbox"/> \$60	<input type="checkbox"/> \$65	<input type="checkbox"/> \$75

- (1) Includes copy of proceedings plus one lunch  
 (2) Includes copy of proceedings plus two lunches, one banquet  
 (3) Includes copy of proceedings plus three lunches, one banquet

Full Time Students—\$10 before May 5th, \$15 after May 5th. Includes copy of proceedings

No. of banquet tickets at \$7.00 each \_\_\_\_\_

No. of people attending spouse's program \_\_\_\_\_

## FORM

Name \_\_\_\_\_  
 Organization \_\_\_\_\_  
 Address \_\_\_\_\_  
 City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

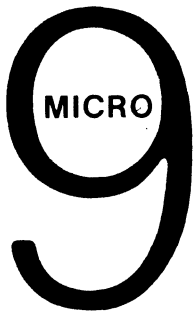
COMPLETE AND RETURN THIS FORM WITH YOUR CHECK TO:  
 (Payable to Utah State University)

Prof. E. E. Underwood  
 UMC 41  
 Utah State University  
 Logan, Utah 84322

FOR LOCAL ARRANGEMENT INFORMATION, CONTACT:

Prof. W. I. Fletcher  
 Electrical Engineering Dept.  
 Utah State University  
 Logan, Utah 84322





---

CALL FOR PAPERS

---

**MICRO-9**

**Ninth Annual Workshop on Microprogramming  
September 27-29, 1976  
New Orleans, Louisiana**

**WORKSHOP FORMAT**

The purpose of this workshop is to bring together practitioners and theoreticians from industry, government, and academia who are interested in problems relating to the underlying concepts and use of microprogramming. Of the papers accepted for publication in the proceedings, only a selected subset will be formally presented during the workshop. In order to assist in the exchange of ideas and the sharing of experiences, most of the time will be dedicated to short informal presentations, discussion groups, and case study tutorials associated with a particular topic. As these sessions will be organized around papers accepted for publication in the proceedings, the importance of these papers cannot be overstressed. Copies of the proceedings will be sent in advance.

**SUBMISSION OF PAPERS**

Authors are invited to submit original papers on recent and novel developments in all aspects of microprogramming. Topics typically covered will include, but will not be limited to, the following:

- high level language support via firmware
- operating systems support via firmware
- performance measurement and evaluation
- system architecture
- microprograms: coding and optimization
- hardware/firmware/software tradeoffs
- teaching about microprogramming
- fault diagnosis and recovery
- microprogramming applications

Preliminary copies of papers, not to exceed twenty double spaced typewritten pages, should be submitted by May 15, 1976. The final papers will be due August 1, 1976. The manuscripts should be sent to:

Professor Peter Kornerup  
MICRO-9 Program Chairman  
Computer Science Department  
University of Southwestern Louisiana  
Box 4-4330  
Lafayette, Louisiana 70504  
(318) 233-3850 Ext. 538

**LOCATION**

The workshop will be held at the Delta Towers Hotel in downtown New Orleans. Within easy walking distance of the hotel is the historic French Quarter famous for its scenic charm, Dixieland jazz, and Creole cuisine.



**Bell Laboratories**

600 Mountain Avenue  
Murray Hill, New Jersey 07974  
Phone (201) 582-3000

Dr. Charles A. Rose, Chairman  
SIGDA/ACM  
Computing and Information Science  
Case Western Reserve University  
Cleveland, Ohio 44106

February 23, 1976

Dear Chuck:

IFIP W.G. 5.2 on Design Automation held a procedural meeting on February 11 which I attended on behalf of AFIPS. The meeting preceded a three-day conference which I attended for the first day. A proceedings is being prepared which will contain 17 papers plus transcripts of the discussion on each. The material is to go to the publisher (North Holland Press) in May. The other activities covered were as follows:

A directory of CAD people is being prepared by Geoffrey A. Butlin, University of Leicester. It seems skimpy on USA people and I told him so. For example, Roy Russo and Chuck Radke were missing. He will send me a copy for review and I will try to get critical colleagues on the list. At some time in the future I may ask for some money to cover mailing the list to USA and Canada entrants.

A glossary is almost ready to be published. It has been reviewed by many people, including myself. The glossary will most likely be a section in the next issue of IFIPS' "Guide to Concepts and Terms in Data Processing."

The W.G. membership will be pruned to eliminate inactive members. If you would consider the matter, I would like to suggest you designate one other person to join me as a representative on the W.G.

The chairmanship changed hands from Jakob Vlietstra of Philips at Eindhoven to Ernest Warman of Perkins Engineers at Peterborough, U.K.

Vlietstra is on the IFIPS 77 (August 8-12 at Toronto) Program Committee. He has arranged for a full parallel sequence of sessions on CAD. That amounts to up to 60 papers and could put a strain on our DAC in 1977. There will also be panels and birds-of-a-feather sessions!

Two books are being prepared by W.G. members:

- 1 - A CAD survey by J. Hatvany (Hungary), W. Newman (USA), and M. Sabin (U.K.).
- 2 - A CAD Handbook by J. Hatvany and M. Sabin.

The publication schedule is unclear.

An invitees-only meeting is being planned for September 29-October 1, 1976, in Hungary, for in-depth discussions of data bases, methods for CAD systems, graphics, skeletal structures of CAD systems, and impacts on users (economic, operational). More will be available on this meeting. Similar meetings on other topics are being considered for 1977 and 1978. These are not full-fledged meetings with proceedings but are more like our workshops.

I will continue to keep you informed.

Charles W. Rosenthal

MH-8624-CWR-ak

Under the umbrella of the International Federation of Information Processing Societies there is Working Group 5.2 on CAD Systems. The Group consists of representatives of national computer societies. I have represented AFIPS since the formation of WG 5.2 in 1973.

WG 5.2 concerns itself with getting people together from all over the world to help disseminate information on CAD. The coverage is of systems (hardware and software) and of applications to many disciplines (electronics, chemical plants, structures, medicine, machining, etc.). The group sponsors work on books, glossaries, and directories of members. It organizes: small meetings of specialists in one phase of the field; invitees-only meetings for in-depth discussions; and, general meetings in connection with large conferences.

On February 12-14, WG 5.2 sponsored a triennial meeting at the University of Texas at Austin. I attended the first day and read the entire draft proceedings of 17 papers. The final proceedings will be published with transcripts of the extensive discussions that followed each paper's presentation. A list of the papers' titles is attached. What follows is a brief review of the papers that impressed me the most. If you are interested, I can send a copy of any paper identified for me.

The Design of Systems for CAD - An interactive, stand-alone, minicomputer system for chemical plant design. Starts with major equipments located and interconnection piping (connectivity) designated. Helps in selecting and routing piping. Uses library of components, large data base (1-10 M words), contains features for checking on and avoiding violations of design rule constraints. Data base seems to be of DMS variety. Use is made of transition networks to specify user commands. In their presentation, the designers displayed a lot of down-to-earth appreciation of CAD systems.

Modeling by Computer - Suggests greater reliance upon procedural (equations or algorithms) representation of objects in data base rather than numerical tabulations. This is to: reduce amount of data space taken; retain characteristics of objects and allow parametrization; and, avoid some problems in rendering instances of objects. Some of our data bases already employ procedures but more use might be made especially as we try for greater independence from processing programs and output devices.

Interactive Graphic Hardware for Practical Use in CAD - The author proposed a display which he judges to have all the characteristics necessary to satisfy CAD needs. There was lively discussion showing very clear lack of consensus except for: more work on ergonomics so as to better understand the assignment of functions to men and machines for different applications; and, on efforts to reach common definitions of the display parameters that are in use.

Relational Data Handling Techniques in Computer Aided Design Procedures - In recent years there have been proposals to use relational data bases as an alternative to hierarchical and network data bases. This paper, too briefly describes a relational data base and its application to a CAD system for civil engineering. For the essential information it would be necessary to consult the references.

A Software System for Computer Aided Activities - Starts with APL and adds to and modifies it to enhance string manipulation and graphics operations.

Artificial Intelligence in Computer-Aided Design - The thesis is that CAD consists of too much implementation of cut-and-dry techniques and too little implementation of true problem solving methods. The latter are supposed to be available from recent work in artificial intelligence. The author told me he expects to have his ideas in operation on the PDP-10 at Stanford this summer. Time will tell!

The REGENT System for CAD - A software system to help generate CAD application programs for diverse fields (fluid dynamics, 3D graphics, flowcharts). REGENT provides tools for adding application oriented statements to the PL/I base and for module generation and management, dynamic array management, data base management and error message handling. REGENT is a basis for integrating all CAD projects under development in Germany.

Raster-Scan Graphics in CAD - Reviews the advantages and disadvantages of raster scan graphics. Advantages cited are the availability of color gray, scale and shading. Disadvantages are poorer resolution, quantization efforts and slower modification speeds than for direct access graphics.

  
Charles W. Rosenthal

### TABLE OF CONTENTS

#### Part I - EXECUTIVE SYSTEMS

- 2/4 (14) Techniques for Processing Interactions in Fortran  
Geoffrey A. Butlin\*
- 2/3 (10) A Software System for Computer Aided Activities  
Mamoru Hosaka, Takeshi Matsushita, Fuminiko Kimura,  
and Naotake Kakishita\*
- 2/2 (5) A Way to Computer Supported Systems for Integrated  
Design and Production Process Planning  
F. L. Krause\* and V. Vassilakopoulos
- 2/4 (13) Artificial Intelligence in Computer-Aided Design:  
The "Tropic" System  
Jean-Claude Latombe\*
- 2/2 (1) The Design of Systems for CAD  
R. G. Newell\*, T. L. Sancha, R. M. Williamson,  
and J. O. Hiles
- 2/3 (9) The Regent System for CAD  
K. Leinemann and E. G. Schlechtendahl\*

#### Part II - COMMAND LANGUAGES

- 2/4 (15) A Text Analyser and Text Generator for Interactive  
Command Languages in CAD  
Ketil Bjø\*
- 2/4 (17) Design Automation Systems (With Special Reference to  
Discrete Microelectronic Units)  
M. A. Gavrilov\* - Absent
- 2/3 (11) High Level Non-Graphical Interaction in Computer-Aided Design  
✓ John S. Gero\* and Warren G. Jullian
- 2/2 (2) The Possibility for the Automatic Production of Command  
Languages  
E. A. Warman\*

Part III - DATA STRUCTURES

- 2/14 ⑬ ✓ Data for Health Building  
R. H. Goodman and M. A. Meager\*
- 2/12 ③ ✓ Modeling By Computer  
Martin E. Newell\* and David C. Evans
- 2/13 ⑦ ✓ Application of Suitable Data Structures for  
CAD in Machine Tool Industry  
W. Eversheim, D. Pfau\*, and R. Rothenberg
- 2/13 ⑫ ✓ Relational Data Handling Techniques in Computer  
Aided Design Procedures  
Giorgio Valle\* - Absent

Part IV - HARDWARE

- 2/12 ④ ✓ Interactive Graphic Hardware for Practical Use in CAD  
J. Hatvany\*
- 2/13 ⑧ ✓ Interactive Graphics as a CAD Tool at Mullard Research  
Laboratories  
D. J. Burnett, R. W. Clarke, B. M. Jones\*, H. R. Sethi,  
and J. A. Weaver
- 2/12 ⑥ ✓ Raster-Scan Graphics in CAD  
William M. Newman\*

---

\* Denotes author that will make the presentation about  
the paper.

---

\*\*CALL FOR PAPERS\*\*

SPECIAL ISSUE ON THE APPLICATIONS OF  
COMPUTER HARDWARE DESCRIPTION LANGUAGES

to be published in  
COMPUTER during 1977

Papers on the applications of CHDL (Computer  
Hardware Description Languages) in the following  
areas will be considered:

1. Fault diagnosis and test generation
2. Computer-aided logic design
3. Hardware compiler (translator)
4. Register transfer level simulation
5. Automatic micro code generation
6. Partitioning
7. Other applications of CHDL

Papers reporting experience in using CHDL as well  
as dealing with future prognosis are also welcome.

You are invited to submit three copies of the  
paper in 15 to 25 double-spaced typewritten pages  
to:

Professor Stephen Y. H. Su  
Guest Editor, COMPUTER  
Department of Electrical Engineering  
Utah State University  
Logan, Utah 84322  
Telephone (801)-752-4100, extension 7806

DEADLINE for submitting papers: November 1, 1976

RECENT PUBLICATIONS.

compiled by

W.M. vanCleemput  
Digital Systems Laboratory  
Stanford University  
Stanford, California

1. DIGEST OF THE 11th IEEE COMPUTER SOCIETY CONFERENCE, WASHINGTON, D.C., SEPTEMBER 1975  
A Rationale for the Random Testing of Combinational Digital Circuits (John J. Shedletsky) (pp. 5-8).  
Fail-Safe Circuits: A Means to Improve Reliability and Maintainability of I/O Subsystems  
(Alan M. Usas) (pp. 9-12).  
MPL-85: A High-Level Microprogramming Language Compiler (C. J. Lane and M. R. Ito) (pp. 49-52).  
Design for Fault Tolerant MOS Memories (Len Levine and Ware Myers) (pp. 97-102).  
An Integrated Hardware/Software Microprocessor Based Development System (D. R. Deuel and J. W.  
Gault) (pp. 163-165).  
Calma's GPL™ Language: A Programming Language for Custom, Turnkey Graphics Systems (Carl Smith)  
(pp. 231-233).  
Software Organization for a Multipurpose, Digital, Design Automation System (Robert J. Smith II,  
and Mathew N. Matelan) (pp. 253-255).  
An Architectural Development Tool for an Intermediate Language Machine (ILM) (Anthony Carpino)  
(pp. 265-267).

2. DIGEST OF THE 12th IEEE COMPUTER SOCIETY CONFERENCE, SAN FRANCISCO, FEBRUARY 1976.

Design Systems - A Five Year View (Charles Rose) (pp. 190-193).  
The Next Stop in Testing and Modeling (S. A. Szygenda) (pp. 194-196).  
On Software Certification (Ralph E. Keirstead) (pp. 222-224).  
Procedure Extraction for Automatic Restructuring (Guy de Balbine) (pp. 225-228).

3. PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE AND EXHIBITION ON COMPUTERS IN ENGINEERING AND  
BUILDING DESIGN, IMPERIAL COLLEGE, LONDON, ENGLAND, MARCH 1976. Available from: IPC Science  
and Technology Press, IPC House, 32 High Street, Guildford, Surrey, England GUI 3EW.

Building Design (Part 1)

Computer-aided Handling of Briefing and Design Information for Building Projects (D. G. Campion).  
The Use of 3-D Interactive Graphics to Aid the Design and Layout of Equipment (C. Johnson, P. Purcell  
and P. Sampson).  
The Inter-relationship of Design and Performance Variables (A. H. Bridges).  
Dynamic Programming in CAD of Buildings (J. S. Gero).  
CEDAR 3: The Philosophy of Basic Software for Computer-aided Building Design and Ensuing Software  
Design Considerations (D. J. Charlesworth).  
New Developments in the OXSYS System (P. N. Richens).  
Introducing Computer Aids for a Department of Architecture in Local Government (R. Th'ng).

Civil Engineering and Plant Desing (Part 1)

A Review of Civil and Process Plant CAD (K. M. Vine-Lott)  
The Use of Computers in the Design and Analysis of a Contractors Work in Connection with the Construction  
and Load out of a 14000 Ton Oil Production Platform (C. J. Riley and D. J. Rolton).

Techniques (Part 1)

Computer-aided Function Allocation and Evaluation Systems (CAFES) (D. C. Whitmore).  
The Design of Interactive Systems for CAD (A.P. Armit and W. S. Elliott).  
Portability and Adaptability of CAD Software (R. L. Schiffman and D. M. Jubenville).  
A Generalized Interactive Three-dimensional Input System (P. R. White and R. E. Garrett).

Design Linked to Manufacture

CAD and the Total Manufacturing Process (M A. Sabin).  
Geometric Design Using a Minicomputer (G. E. G. Bishop).

A Computer System for the Design and Manufacture of Aircraft Components (P. Aughton)  
POLYSURF: An Interactive System for the Computer-Aided Design and Manufacture of Components  
(A. G. Flutter and R. N. Rolph).  
A Computer-aided Design System for Pattern Grading and Marker Making in the Garment Industry (U. Cugini,  
A. Catastini, C. Cavagna and P. Moro).  
An Interactive Computer Graphics Approach to the Problem of Netting of Plane Parts (J. Øian,  
B. Hasselknippe and F. Lillehagen).

#### Mechanical Engineering (Part 1)

Review of CAD in Mechanical Engineering (B. Gott).  
Designing by Simulation (E. A. warman).  
Fiat's Automated Mechanical Drafting System (V. Moreggia and R. Tavan).  
A System for Describing Mechanical Engineering Components (D. T. Stevens).

#### Structural Engineering (Part 1)

Current Structural Analysis Systems and Development Trends (G. J. V. Shoppee).  
A Computer-aided Finite Element Idealization and Mesh Generation System (S. Singh).  
MINT: An Interactive Finite Element Mesh Generator for Tubular Joints (I. A. Nazlawy).  
Interactive Selection and Presentation of Results from a Substructuring Analysis System (H. A. Nasreldin  
and A. G. Young).  
The Development and Usage of a Computer Program Suite for the Static and Dynamic Analysis of high  
Pressure High Temperature Pipework (J. A. Bennett).  
ASAS -- Progress Toward Acceptability (N. C. Knowles).  
Social Aspects of CAD (M. J. E. Cooley).

#### Building Design (Part 2)

Databases for Physical System Design: A Survey of US Efforts (C. M. Eastman).  
A Graphics System for Building Design in the Canadian Government (J. K. Robertson).  
CAD in the Design of Heating, Water Supply and Air Conditioning of a Public Office Building (M. Hakkila).  
LUMPS (Land Use Master Plan Substitute (P. Mugnaioni and R. A. Goddard).  
User Aspects of SAR 70, an Interactive System for the Generation of Alternative Dwelling Layout  
(J. Amkreutz and P. J. M. Dinjens).  
Data-structure for Functional and Geometrical Description of Buildings (K. Agger).  
Social Aspects of CAD (M. J. E. Cooley and T. W. Maver).

#### Civil Engineering and Plant Design (Part 2)

The Dynamic Straight Line Data Handler and the Virtue Balance as Process Plant CAD Aids (M. E. Leesley,  
G. Heyen and D. D. Mulraney).  
STEM: Heat Exchanger Design/Plant Design (P. Cavallo).  
A Modern Approach for Structural Design and Pipe Culverts (M. G. Katona and J. M. Smith).  
CABLES: An Interactive Design System for Cable Layout (P. Herke and B. Hopson).

#### Informal Papers

The Easy Way in to Computer-aided Building -- via the Design Office Consortium (R. Howard).  
SESAME -- A Second Generation Suite of Programs for Structural Engineers (C. Chapman).  
Experience with CAD for Steel Framed Structures (S. H. Simmonds, A. M. Lount, and K. M. Vine-Lott).  
A CAD System for Building Structures in 'BUILDING RC/SRC/S' (M. Horii, A. Wada, T. Aoyagi and F. Suto).  
A UK Development in Interactive Graphics Systems (J. R. Cookson).  
The DISSPLA Graphics Software Package (M. Repko).  
GINO-F: A Graphics Standard? (P. Lever).  
An Integrated Minicomputer Graphics System (R. K. Oatman).  
MOSS -- Modelling Systems for Geo-coded Information (E. Malcomson).  
Interactive Computer-aided Design and the DECIDE System (A. W. Beeby and H. P. J. Taylor).

#### Mechanical Engineering (Part 2)

CAD in the Context of Engineering Design - The Designer's Viewpoint (D. G. Smith and S. Pugh).  
ASKFOR: A Design Analysis Aid for Bearing Applications (B. Greener).  
The Computer-aided Design of a Mechanism Using Standard Computer Programs (D. G. Wilkinson).  
Development of a Mechanical Design by a Method of Synthesis Using Basic Elements (E. Muller).

#### Structural Engineering (Part 2)

Beam Details for Steel Framed Buildings (D. M. Brotton and M. G. Moore and F. Glover).  
Computer Graphic Techniques in Structural Optimization (A. S. Sambura).  
The GENESYS System (I. H. Hunter).

#### Techniques (Part 2)

An Overview of Interactive Graphics for CAD (D. J. Humcke).  
Criteria for Interactive Techniques in CAD (J. L. Wilson).  
Computer Modulated Drawing in Graphics Design (C. M. Williams).  
Models, Data-Structures and Storage Structures (D. F. Barnard).  
Three-dimensional Digitizing Techniques for Computer-aided Animation (C. B. Besant, A. Jebb, and C. Yi).  
Computer Generated Stereograms for Design Analysis (R. G. Wollacott and A. G. G. Finney).  
GMB: Program System for Interactive Graphics (E. Hoerbst, M. Gonauser and J. Weiss).

## A MODEL OF REAL TIME CONTROL SYSTEM PRODUCTION

M. N. Matelan

Design Automation Project  
Engineering Research Division L-156  
Lawrence Livermore Laboratory  
Livermore, California 94550

### Abstract

Many facets of Computer Science and associated technologies may be profitably viewed as dedicated real time control activities. Production of systems to exercise such control has been difficult and costly. An abstract model of the process of producing these systems is presented. The model indicates three areas of the design problem amenable to automation: 1) the selection and configuration of hardware; 2) the production of software; and 3) the selection of a monitor to maintain real time integrity of the entire system. The concept of hardware binding is introduced and it is shown that delaying the point in the design cycle where hardware is functionally bound allows a new approach to machine independence.

This paper comprises two chapters of a larger work which develops an implementation-independent application-specification language and techniques to automate control system design. Such an automation system is intended to produce a hardware configuration listing and software which together define a complete dedicated real time controller. Development of this system is based on the model presented here and is an on-going design automation project at Lawrence Livermore Laboratory.

### Acknowledgment

This research was performed under the auspices of the U.S. Energy Research and Development Administration under contract #W-7405-ENG-48, and in conjunction with the Department of Computer Science, Southern Methodist University.



"Our life is frittered away by  
detail. ... Simplify, simplify."

- Henry David Thoreau  
Walden, 1854

## CHAPTER I

### INTRODUCTION

The use of digital computers as real time monitor and control devices is an increasingly important aspect of computer technology [Schoeffler, 1972]. Further, many computer system functions (both hardware and software), not traditionally associated with the engineering discipline of process control may be cast in terms of controlling: correctly responding in a time-critical manner to a set of unpredictable but well-defined conditions. The necessity of imposing strict timing constraints on this definition of control stems from the very nature of real time problems [Opler, 1966]. The design of systems (composed of both hardware and software elements) capable of exercising control in a real time environment has proven to be a very complex and expensive undertaking. It is the purpose of this research to develop concepts which reduce that complexity and methods which allow automated production of real time control systems.

### REAL TIME DEFINED

A precise definition of the term "real time" which is acceptable to the computer science community at large apparently does not exist [Barkley, 1975]. Since the concepts of real time action and reaction are of central importance to this research, however, a working definition is necessary. The primary attribute of a real time situation is the existence of timing constraints so rigid that statistical approaches are not acceptable. Such a situation has been called "hard" real time, as opposed to the "soft" real time typical of interactive and time-sharing systems [Manacher, 1967].

Real time also involves the idea of controller subservience to environmental conditions which are external to it, which exist in the "real" world and which have critically time-bounded sense and response requirements. The key factor in determining the domain of the "real" world (encompassing those entities which require rigid timing constraints) is reproducibility. Information that is not reproducible is by definition lost forever. Real time control applications are critically sensitive to lost data.

In order to formalize these intuitive constituents of real time, two views of time are necessary: physical time and abstract time. The arrow of time in the physical sciences never flies backward, but always and inexorably forward [Layzer, 1974]. Therefore, conditions which are a function of physical time may never be reproduced exactly since the time coordinate is constantly increasing [Einstein, 1921]. Contrast this to the concept of abstract time where the flow of time is a fluid which can be reversed, halted, or varied in rate. Abstract time is essentially a method of marking transitions in a state space. The progress of a computation as characterized by Dijkstra's textual and dynamic indices occurs in abstract time [Dijkstra, March 1968]. Given a well-formed (i.e., deterministic) program, the same sequence of states will occur each time the program is executed regardless of the physical rapidity of the execution. This ensures reproducibility of results regardless of when, in physical time, the program is executed. The relationship of abstract to physical time and the importance attached to lost information may be used to determine whether a particular situation is real time in nature.

For example, interactive computing has often been considered in some sense to be a real time problem. When a user inputs a command to an interactive system, the controller responds following an interval of physical time. However, if the input were lost, it could be retyped without loss of information. Also, the time constraint on response is usually couched in terms of "acceptable" response which may be, and is, studied in statistical terms [Coffman and Denning, 1973]. It is acceptable for an occasional response to be unreasonably long. Here, abstract time is related statistically to physical time constraints, and lost

data (while annoying) is not serious.

Consider now an application which is of the "hard" real time type: a digital controller used to monitor the progress of a chemical experiment [Fisher, 1974]. If the abstract time of the controller cannot meet the demands of the physical time constraints of the experiment, data is lost forever. Moreover, a catastrophic reaction may occur. In the real time situations considered in this research, physical time constraints are of paramount concern; data must not be lost and response must be guaranteed.

Thus, an ontology of real-time control defined by sensitivity to irretrievably lost information and a concomitant need for strict timing considerations allows the province of real time computing to be bounded.

Real time situations are those in which the abstract time interval to the controller must be strictly bounded by physical time constraints associated with irreproducible events in the controlled environment.

It is this interpretation that will be implied by the use of the term "real time" throughout this work. A system in which all activities are real time in nature, sometimes called a pure real time system, will be referred to simply as a real time system. These real time systems have been primarily used in special purpose, or dedicated applications such as control, instrumentation and communications [Korn, 1975]. A dedicated real time system is a pure real time system whose design is determined by a single application so that its configuration remains relatively fixed throughout its service lifespan. While the use of such systems has until recently been limited to a few areas of computer science, it will be shown that they can be expected to play a much larger role in the future.

#### THE EVOLUTION OF REAL TIME SYSTEMS

Until recently, the study of real time computer-based systems received its impetus from problems encountered in applying digital techniques to the area of automatic control. Automatic control is that discipline

concerned with controlling the operation of a system (biological or mechanical) according to given criteria without human intervention [Aizerman, 1958]. The central issue of early automatic control was the use of feedback to maintain a variable or set of variables in some optimal condition. The design of linear servomechanisms and development of a theory of optimal filtering and prediction marked the advance of automatic control theory [Hermes, 1969]. Design techniques centered on modeling the process under control as a system of continuous functions. Once a description of the process was available, analog controllers were designed to maintain the value of each function within some optimal operating range.

The introduction of digital computers to automatic control in the late 1950's marked the beginning of computer process control, a much broader and more complex form of control [Lee, Adams and Gaines, 1968]. A central digital computer could be used to replace a number of local analog controllers and still perform such functions as error condition monitoring or record keeping. The usefulness of digital computation in automatic control is witnessed by the large share of digital equipment being dedicated to it [Bowers, 1975]. Hybrid systems composed of local feedback loop regulators monitored by digital controllers are also in use. The growing use of digital computers has led in turn to an increasing need for new control system design techniques. The replacement of automatic regulators with digital controllers gave new importance to the simulation of analog methods useful in representing dynamic continuous processes. Dynamic programming was applied to problems of optimal control [Boudarel, 1971] as were discrete time methods [Cadzow, 1970]. Early activity in defining design methods for process control systems was primarily concerned with analyzing the process to be controlled rather than the integration of the controller, but the problems of producing control programs were quickly apparent.

#### PROGRAMMING REAL TIME SYSTEMS

There are two basic software elements in a real time control system: the tasks and the executive. Tasks are individual procedures (often

called drivers) which together define responses to situations in the controlled process required of the controller to maintain regulation. The executive is a program which must coordinate the execution of the task group so that each situation elicits an appropriate response.

The executive is a form of multiprogramming scheduler, the analysis of which is treated in operating system theory. However, most of the work done in the area of multiprogramming involves a statistical approach appropriate to time-sharing systems. Non-statistical approaches tend to examine methods of assigning non-repetitive tasks to multiprocessor systems [Manacher, 1967]. Moreover, multiprogramming methods and research are heavily weighted toward consideration of large, expensive computing systems not usually appropriate for use as dedicated real time controllers. The expense of large systems dictates that they be used to their fullest capability; that is, a major concern of large system multiprogramming methods is the maximization of machine utilization.

Even those papers concerned with hard real time scheduling orient their work toward maximizing hardware utilization [Liu and Layland, 1973]. This orientation in multiprogramming scheduler research is directly counter to the realities and current needs of dedicated real time systems. The rationale for using large centralized computer systems is economic; throughput seems to increase more than proportionally with system cost [Grosch, 1953]. Thus, large general purpose systems must be configured to meet a multitude of processing demands which can only be anticipated in terms of statistically modeled job mixes. In the early days of computer process control, the high cost of hardware motivated a similar emphasis on maximizing utilization, resulting in the addition of so called background tasks not directly related with the controlled process. However, as hardware costs began to drop, the orientation and goals of process control software development diverged from the mainstream of operating system theory which primarily centered on large general purpose systems. Introduction of the relatively inexpensive mini-computer in the mid-1960's dictated further specialization of process control software by allowing dedication of a processor to a specific application. The costs associated with software development for these systems were often the principal factors in both cost and time overruns.

This problem also arose in large system development, but the high initial cost of hardware tended to mask the long-term costs of software. The current interest in structured programming and software engineering reflect an awareness today of the true (that is to say expensive) nature of system software development.

The advent of the microprocessor has reversed the original cost positions held by hardware and software in early process control system design. Processing hardware powerful enough for many applications is now so relatively inexpensive that the use of traditional, large system multiprogramming techniques in connection with dedicated real time systems can be self-defeating. Unfortunately, most designers are either using traditional multiprogramming monitor approaches or are reverting to the practice of machine code programming [Biewer, 1974]. Today, software production for dedicated real time systems is commonly the most critical aspect of the system design. Software development techniques geared to the time-critical, deadline-driven nature of dedicated real time systems are clearly needed.

#### THE IMPACT OF MICROPROCESSORS

Like the term real time, the term microprocessor has not yet been defined in a way acceptable to most of those who use it. There does, however, seem to be a consensus concerning the eventual benefits that microprocessors will bring to nearly every facet of computing. Most definitions of the term included a combination of packaging and processing properties. The most common notion is probably that of a general purpose central processing unit on a single integrated circuit "chip." Definitions involving rapidly changing technological parameters are doomed to change as rapidly. For the purposes of this work, microprocessor will be defined, in terms which relate the economics of hardware and software, and for this reason would seem to be more stable. Such a definition must reflect the rapidly changing nature of digital hardware.

When all general purpose hardware was expensive and software costs had not yet been identified, hardware was considered to be the unchange-

able base upon which flexible software was layered. As needs changed the software was altered, often being made to absorb by complex construction those changes which logically should have been made in hardware (had the cost not been prohibitive). Software was thought to be, in a sense, plastic. System design usually began with hardware development and progressed to software considerations (sometime as an afterthought). Occasionally, someone would mention that hardware and software should be designed concurrently. Inexpensive hardware (reflected in a multitude of processor architectures) allows this to be done by changing the nature of the design problem. The plasticity of hardware is today approaching that plasticity always apparent in software.

Therefore, the term microprocessor will be used in this work to connote general purpose computing hardware so inexpensive that hardware costs and fixed architectures are no longer the primary design constraints. Instead, hardware becomes a design element bending to meet those requirements best suited to a hardware realization.

This view of hardware allows a new latitude in system design in general, and especially in dedicated system design. However, this new flexibility is not being widely exploited. Designers of dedicated systems still think of the microprocessor as a small computer, which implies a data processing role. As done in the past, the hardware is designed, the microprocessor is bound to a specific configuration, and finally the software is written, usually in assembly language. Programs produced in this way are quite hardware dependent and often needlessly complicated by hardware idiosyncrasies. The bias toward designing systems around a central computer as opposed to designing with computers as design elements is largely cultural; it is a bias rooted in an economic situation that is no longer relevant in many applications.

#### REDUCING THE COMPLEXITY OF REAL TIME SYSTEM DESIGN

The synthesis of digital systems may be viewed as requiring two general tasks:

- 1) the selection and configuration of hardware, and
- 2) the production of software.

The order in which these tasks are typically accomplished reflects the traditional hardware-then-software approach taken when producing a system design. The availability of microprocessors, and the realization that real time control problems are especially suited to them, allows a new approach to be considered. (The term system, as used here, is meant to include both hardware and software constituents needed to effect control in accordance with requirements of an application.) The approach has as its goal an overall reduction in the complexity associated with producing a dedicated real time control system. Design complexity may be reduced in at least two ways, both of which impact the process of expressing a problem.

First, the removal of all processing hardware considerations from the description of an application would reduce overall complexity. Limiting hardware considerations to those found in the controlled environment forces attention to the problem definition and away from its realization (the controller). Second, complete elimination of the need to explicitly define the concurrent timing requirements of the several control functions would further reduce design complexity. Obviation of multiprogramming considerations allows the designer to partition the problem into functional sub-problems, as if there were available a separate processor for each function. Both of these approaches will be examined.

#### HARDWARE BINDING

It has often been observed that programming complexity is reduced by increased levels of machine independence. However, true machine independence is difficult to achieve. One of the main sources of this difficulty is the traditional economic necessity of viewing hardware as fixed and software as changeable. In system design, software production usually begins after the hardware has been selected and at least partially configured.



The reality of plastic hardware suggests that it is no longer necessary or appropriate to fix the hardware configuration at the beginning of the design process. Selection of hardware components may be made at various points in an evolving design. A powerful software concept used to describe the fixing or associated of a symbolic reference to a specific programmatic entity is that of information binding [Elson, 1973]. That point in time (relative to some time reference frame) when a symbol is bound to an entity is often called the binding time of the symbol. No corresponding term which denotes the concept of associating various application requisites to particular hardware elements is in common use. This author proposes that the phrase hardware binding be used to convey this concept. Thus, the points in the design cycle when various required hardware components are firmly assigned to realize specific functions associated with the application, can be referred to as hardware binding times.

One way to achieve machine independence then, is to delay all hardware binding until after problem definition time (analogous to the program creation time of information binding). Delaying hardware binding until after the problem has been described by the designer means that the application description can be completely machine independent and transportable. Indeed, it will be a description of the problem only, with the actual hardware configuration to be used being deferred to a selection and assignment process later in the design cycle.

#### REDUCING TIME-RELATED COMPLEXITY

Human beings usually find static relations less difficult to deal with than dynamic ones. Reducing complexity caused by the use of static descriptions (programs) to define dynamic processes is a major goal of structured programming [Dijkstra, March 1968]. Since the static representation of a real time problem must define a time-bounded dynamic process, producing correct real time programs is even more difficult than programming problems not directly dependent on physical time. This intrinsic complexity is usually magnified because of the multi-functional nature of real time controllers.

Much of the power inherent in the use of digital computers as controllers derives from their ability to monitor and respond to more than one application parameter. A single processor involved in controlling more than a single function is said to be multiprogrammed. Multiprogrammed systems require a process to allocate system resources to subfunctions in a way that satisfies some predefined, time-related criterion relative to all subfunctions. The allocation process is usually called a scheduler. In the case of real time control, the scheduler criterion is the strict adherence to the hard deadlines and repetition rates associated with the control tasks. Relatively little work has been done on the properties of time-critical scheduling [Serlin, 1972; Manacher, 1967; Liu and Layland, 1974].

The necessity of multiprogramming in control programs adds greatly to the difficulty encountered in their production. Removal of the necessity for explicitly programming or selecting a scheduler would significantly reduce the complexity associated with securing a control process from a problem definition. This aspect of producing control software is especially important in connection with microprocessors. It is not likely that they can economically support the "universal" schedulers sometimes supplied with more expensive hardware.

#### AUTOMATING ASPECTS OF SYSTEM DESIGN

The research described in this dissertation combines the notions of delayed hardware binding time and removal of explicit scheduling considerations to significantly reduce the effort required to develop a control system given a well-defined problem. The power of the combination greatly exceeds the capabilities of methods incorporating only one of these elements. The approach includes three aspects of system design:

- 1) Problem Specification,
- 2) Selection of Hardware, and
- 3) Production of Software.

It is the goal of this research to simplify the first aspect and completely automate the remaining two.

The machine independence afforded by delaying the selection and configuration of hardware until after problem description, coupled with the removal of scheduling considerations means that the designer need only specify the behavior of the proposed control system. This means the designer must express only what the system is to do (i.e., behavioral description) rather than how the final system constituents are to do it (i.e., implementation prescription). Descriptive tools of this sort have been termed very high-level languages [Tucker, 1975].

Once the behavior of the needed controller has been described, a system capable of analyzing the description would be invoked to produce both a complete control program and a configuration specification for the hardware that is to execute the program. The production of software in this way has been called automatic program writing [Lee, 1968]. The selection and configuration of hardware will be termed automatic hardware binding.

#### APPLICATION AREAS

Although the major use of dedicated real time control systems has been in process control, many diverse areas of computer science could also benefit from a hard real time approach to traditional problems. Brief examples from operating system design, computer architecture and logic design indicate the applicability of these ideas.

The technique of layering in operating system design, from an outer (user-directed) layer to an inner (machine-directed) layer, has many advocates [Dijkstra, May 1968]. It has been noted that to work cleanly, the layers must be partitioned dynamically as well as functionally [Lynch, 1972]. The lowest layers in an operating system contain many more functions relating to physical timing constraints than do outer layers [Randell, 1969]. Lower levels have been characterized by the short term nature of the policies which guide their implementation; duration of actions at these levels are usually discussed quantitatively in terms of the speed of the host hardware [Brinch Hansen, 1972]. The functions of these lower levels lend themselves to representation as

hard-real time control sequences.

The availability of microprocessors for dedication to selected support (lowest layer) functions of an operating system would have a great impact on the organization of the remainder of the system. The elimination of expressly programmed multifunction scheduling and timing could enhance transportability. An essentially machine independent operating system design could instead rely on "blackbox" control subsystems to mate the upper levels to the host hardware.

In the area of computer architecture, microprocessors are being aggregated to form distributed processor computing systems called multi-micro-processors [Fuller, Siewiorek, Swan, 1976]. Two important problems which must be solved when using this approach are the production of local control programs to synchronize the system, and the configuration of each node (consisting of microprocessor hardware) so that intercommunication is possible. The collection of valid interactions allowed in the aggregation may be viewed as a communication protocol observed by each node. A hard real time control approach to the management of networks of processors appears to be very appropriate [Matelan, August 1975].

While the effects of microprocessors on conventional process control, operating systems and computer architecture are evolutionary (being an extension of the use of computers in a more dedicated way), their effect on logic designers is revolutionary [Rathmuller, 1975]. The basis for the rapid acceptance of the microprocessor as a logic design element is economic. Many different logic functions can be simulated by a general purpose computer--making it, in essence, a universal logic element capable of performing a number of functions simultaneously. A universal logic element produced in high-volume can economically replace many lower volume special purpose items.

The universality of microprocessors in logic replacement stems from their adaptability. However, they must be programmed before they can simulate any specific logic element they are to replace. The necessity of producing a program has often cost the designer more than was gained in flexibility. Much of the difficulty in using micropro-

cessors is caused by viewing them as small computers. This causes designers to design around the processor rather than design with it, another legacy of the high cost era of computational hardware.

A way to avoid concentration of effort on the intricacies of the processor at the expense of other parts of the design, is to consider the microprocessor a "blackbox" universal element. This could be done by specifying the inputs, outputs, timing and external function of the element. This view can be accommodated by approaching the universal logic design element as if it were a dedicated real time controller.

#### FOCUSING THE RESEARCH

One of the most significant uses of microprocessors will be the replacement of hardwired logic. This use of microprocessors is well adapted to the subject of this dissertation: the reduction of complexity in producing dedicated hard real time control systems. For these reasons, examples illustrating the applicability of various concepts will be focused on the use of general purpose processors for universal logic elements, whenever appropriate.

Results of this research have two distinct aspects: abstract and applied. Initial work centered on developing a formal model of the process of designing real time control systems. Insight gained from the model led to an investigation of methodologies one might use to reduce the overall complexity associated with designing such systems. This dichotomy is reflected in the structure of the dissertation.

Concepts which allow exploitation of delayed hardware binding and reduction of timing complexity are derived from an abstract model of the process of producing control systems. The model is developed in Chapter II. The consequences of the model which allow a structured approach to control system design are the subject of Chapter III. Chapter IV discusses features necessary in a descriptive control system definition language, while Chapter V concerns the description of hardware capabilities needed to realize a design. Chapter VI investigates techniques that would allow complete automation of control system pro-

duction (both the programming and the hardware configuration aspects) given a hardware independent, functional application description. Directions for further research and concluding remarks are found in Chapter VII.

## CHAPTER II

### A MODEL OF REAL TIME CONTROL SYSTEM PRODUCTION

The design of real time systems is often considered an art. An essential property of an art is the inability of its practitioners to specify the process involved with sufficient precision to allow consistent repeatability. This research is directed toward removing various elements of real time control system production from the realm of art. In order to accomplish this, a specification of the process of designing control systems more precise than currently available is required. The complexity of designing actual control systems suggests the development of a simplifying model which abstracts the essential processes and elements of their production. Using insights gained from such a model, techniques have been developed to produce real time control systems in a more consistent (and therefore less prodigal) way.

The complexity of the process under study dictates complexity in the details of its model. Fortunately, the view taken of the process (and reflected in the model) greatly affects how this complexity is perceived. Complex issues cannot be made simple and retain their power, but they may be organized in ways which reduce the complexity of certain components at the expense of others. The simplified perspective afforded by the model indicates a new approach: reduction of complexity encountered in specifying an application at the expense of developing a system capable of automating the production of an acceptable controller. A relatively large number of parameters must be specified if the model is to provide enough detail to form a basis for automation. A hierarchical structure is one of the most natural ways of organizing a volume of information for clear presentation [Simon, 1962]. The detailed structure of the Control System Design Model is well adapted to such an exposition.

The model is divided into three levels of increasing resolution referred to as levels of abstraction [Dijkstra, May 1968]. The highest (most abstract) level is defined by a sextuple which is essentially an overview of the design process. There is then an intermediate level of representation which divides four of the components of the sextuple into two subcomponents each, which reflect a dual view of time. This level acts as an interagent relating generality and specificity. Finally, the lowest level of abstraction (the infrastructure) is defined. It is upon the details of this fine structure that techniques for reducing the accumulative complexity of real time control system design are based.

#### THE HIGHEST LEVEL OF ABSTRACTION

The complexity or simplicity encountered in examining a process is directly and critically related to the notation used to express it. Care must be exercised in choosing the constituents of the model to avoid masking the essential elements involved in producing real time control systems. Much of the complexity encountered in previous approaches to this design problem was due to an inability to separate the problem definition from the realization of a problem solution. Not only did the software of the host machine directly enter the prescription of

the application, but many features of the hardware needed to be specified. Some progress has been made toward use of higher level languages to reduce the level of host machine software familiarity required to define an application. However, no design aid truly removes the burden of specifying hardware-dependencies from the designer. "Machine independence" is a phrase which often denotes the aggregation of machine dependencies in a program. Machine independence as discussed in this dissertation has a more fundamental interpretation: no element of the hardware used to realize a control situation need be stated or implied. The author's desire to derive techniques which allow the design process to approach the ideal of complete host software and hardware independence is reflected in the form and orientation of the model used to represent the controller synthesis procedure.

An abstract system can be described as a set of objects (which may themselves be sets) together with operations combining them [Herstein, 1964]. The model on which this research is based consists of four sets of information (or "specifications") and two transformation rules. The transformations take ordered pairs from product sets (formed from three of the sets) into the fourth set. The Control System Design model (CSD) is formally expressed as a sextuple:

$$\text{CSD} = (\mathbf{A}, \mathbf{R}, \mathbf{M}, \mathbf{C}, \Gamma, \Psi) \quad [2.1]$$

where:

- A** is the set of all possible applications which may be stated in terms of dedicated control. The set **A** is known as the Application Set; a, element of **A**, is a particular application. (The element a is underscored to avoid confusing contexts.)
  
- R** is the set of all possible system implementations which can realize the goals of the Application Set, **A**. The set **R** is known as the Realization Set; r, element of **R**, is a specific hardware/software configuration associated with a particular computing system.



**M** is the set of all monitor strategies capable of temporally controlling a functionally correct realization of an application. The set **M** is known as the Monitor Strategy

Set;  $m$ , element of **M**, is a specific monitor strategy where the term strategy encompasses those policies and algorithms necessary to define a monitor.

**C** is the set of all dedicated control systems which correctly exercise control according to applications, elements of **A**, realized by hardware and software, elements of **R** and **M**. The set **C** is known as the Control System Set;  $c$ , element of **C**, is a particular control system. **C** also contains  $\emptyset$ , the unrealizable control system.

$\Gamma$  is a transformation from the Cartesian product set of **A** (the Application Set) and **R** (the Realization Set) into a subset of **A** x **R**. This subset (when non-empty) contains those  $(a,r)$  pairs in which  $r$  functionally realizes all non-temporal requirements of  $a$ .  $\Gamma$  is known as the Functional Transformation. The subset of **A** x **R** selected by  $\Gamma$  is referred to as the locus of A, written  $G$ . The subset of **A** x **R** selected by  $\Gamma$  for a particular element  $a$  of **A** is referred to as the locus of a, written  $g$ .

$\Psi$  is a transformation from the Cartesian product set of  $G$  (the locus of **A**) and **M** into **C**. **C** (when non-trivial) contains as elements those pairs formed from elements of the locus of **A** and elements  $m$  of **M** which together define control systems that meet both the functional requirements (guaranteed by definition of the locus of **A**) and the temporal requirements of **A** (maintained by **M**).  $\Psi$  is known as the Temporal Transformation.

As there are a number of symbols used to express the CSD model, remarks concerning notation may be useful to the reader. In general, sets which contain diverse information will be symbolized by upper-case

Roman letters, the corresponding lower-case letter representing a particular subset. Upper-case Greek letters symbolize transformations, while lower-case Greek letters stand for sets which can be expressed quantitatively. In cases where upper-case Roman and Greek letters have the same representation, the reader should resolve the ambiguity in favor of the Roman letter (that is as a set rather than a transformation). The symbol  $\Sigma$  will be used in the conventional sense to indicate summation.

The constituents of the Control System Design model are related by a single expression which symbolizes the design process:

$$\Psi: (\Gamma: A \times R) \times M \rightarrow C. \quad [2.2]$$

Elements of  $C$  are specifications of control systems capable of exercising control in a manner consistent with the demands of elements of  $A$ . For a particular application then, we may write:

$$c = ((a,r),m). \quad [2.3]$$

All  $c$  which satisfy equation 2.3 are said to exhibit Systemic Congruence with  $a$ , a particular application.

#### INTERPRETATION OF THE MODEL

The relation  $c = ((a,r),m)$  specifies the result of control system production. The implications of the model in automating much of the design task now done manually by designers is the topic of Chapter III. However, it is useful to relate more concretely the six elements of the model to the process constituents they describe.

The Application Set is defined to be all applications which can be specified in terms of dedicated control. There is no mention whatever of possible realizations of these applications. The set  $A$  should be thought of in the sense of a problem specification rather than a solution prescription. There is no intimation of the processor or software to be used, or the ways in which they are to be combined to form a system. The most significant feature of this model and this research is the pro-

position that a knowledge of realizations is unnecessary in specifying an application. All components (hardware and software) used in realizing the final system must be selected from the Realization and Monitor Sets. Elements of the Application set merely guide component selection; they in no way specify which elements must be included.

Each element of the Realization Set defines a particular amalgamation of hardware and software which, when executing, performs a specific, repeatable task or group of tasks. Each element of the set  $R$  need not be functionally unique, although each element must be a unique hardware-software combination. Restating, two or more elements of  $R$  may be functionally identical (that is, define the same "black box"), but cannot specify exactly the same software for the same hardware. This approach eliminates duplication of realization from  $R$  while acknowledging that there are many ways to solve a problem.

As an illustration, consider the effects on the set  $R$  of a form of the well established hardware-software tradeoff question. The realization of a task can be made in a broad spectrum ranging from a minimum of hardware with complex software, to the exclusive use of hardware. The abstract set  $R$  contains for the task an element for each increment in the spectrum. Further, there is a spectrum of similar variations for each processor. Finally, there are myriad tasks and groupings of tasks if the set  $R$  is to realize any element of set  $A$ .

The product set,  $A \times R$ , comprises all possible pairings of the elements of  $A$  and  $R$ . That is,  $A \times R$  is all applications paired with all realizations. It is obvious that the vast majority of these pairs is useless. Some pairs will have an application and a realization that simply cannot be related. Other pairs will have realizations which very nearly meet the needs of their associated application. Such close, but not quite acceptable realizations can be likened to systems which (from the application designer's point of view) still have errors, or "bugs."

Now consider those pairs  $(a,r)$  in which the functional behavior of the realization is in accord with the needs of the application. In situations where there exist no constraints other than functionality,

any  $r$ , element of  $R$ , can be considered a solution of  $a$ , element of  $A$ . The purpose of a compiler may be analogously described in the sense that it selects a software realization of an application given a predetermined host environment (hardware and operating system). Since the number of pairs in the product set for which  $r$  functionally satisfies  $a$  is very small relative to the entire set, it is useful to give such elements of  $A \times R$  a name. This assemblage is  $G$ , referred to as the locus of  $A$ . The term references only the Application Set to reflect the orientation of the model. Realizations should be selected for their ability to solve the application problem specification, and not contrariwise. The locus of  $a$  is the assemblage of all elements of  $A \times R$  which functionally satisfy the requirements of a specific application. The locus of  $a$  then, would contain those functionally correct portions of the trade-off spectrum for each processor in  $R$ . Functionally solving a particular application problem may be viewed as selecting any member of the locus of  $a$ , and utilizing its corresponding realization as a system. In fact, the model to this point describes systems which are not constrained by physical time requirements (for example, batch systems). The application of  $I$ , the Functional Transformation, generates the locus of  $a$ , any element of which would constitute a solution for such a system:

$$I: A \times R \rightarrow C. \quad [2.4]$$

However, there arise complicating constraints when physical time is the master and abstract time the slave. The production of real time control systems involves more than a functional congruence to the application. A strict adherence to timing constraints is required.

The necessity of temporal congruence between application and control system dictates the inclusion in the system of a dynamic agent charged with enforcing timing constraints. Such an agent is a monitor,  $m$  element of  $M$ . As there are many ways to realize functional congruence, there are many monitors to provide temporal congruence. Monitors are distinguished by the methods and policies they use. These are referred to collectively as the strategy of a monitor. The production of a real time control system then, may be characterized as selecting an element of the locus of  $a$  which can meet the timing constraints of the appli-

cation. The timing constraints are said to be satisfied if a particular monitor strategy exists which can direct the dynamic performance of an element of the locus such that no application function violates its own constraints during worst case system activity. The product set of all elements of the locus of **A** and all elements of the Monitor Strategy Set comprise all pairs of guaranteed functionally congruent realizations of **A** with all monitors in **M**. It is the role of  $\Psi$ , the Temporal Transformation, to map into **C** all those pairs which allow temporal congruence. All non-null elements of **C** must be systemically congruent to some application  $\underline{a}$ , element of **A**, since all elements of the locus are functionally congruent and  $\Psi$  maps only those locus elements which are temporally congruent.  $\Psi$  represents the timing analysis which must be performed on each monitor/realization pair to guarantee that the two aspects of systemic congruence (i.e., functional and temporal congruence) are met by which a control system (c) may be used to implement a particular application ( $\underline{a}$ ).

The transformations  $\Gamma$  and  $\Psi$  may be viewed as isolating viable control systems from the product of all elements of **A**, **R** and **M**. This is represented schematically in Figure 2.1. The outer boundary includes all combinations of **A**, **R** and **M**.  $\Gamma$  must determine an inner boundary containing only combinations of **G** and **M**. Finally,  $\Psi$  must isolate those combinations of **G** and **M** which demonstrate systemic congruence with applications in **A**.

Finally, it is instructive to discuss that subset of **A** which cannot be mapped into a systemically congruent element of **C**. This subset is termed the kernel and maps to  $\emptyset$ , element of **C**. A particular application in the kernel must fail to meet either functional or temporal congruity, or both. First, if  $\Gamma$  cannot choose an ordered pair (a,r) such that r functionally realizes  $\underline{a}$ , then

$$\Gamma: \mathbf{A} \times \mathbf{R} \rightarrow \emptyset \quad [2.5]$$

and thus

$$\Psi: (\Gamma: \mathbf{A} \times \mathbf{R}) \times \mathbf{M} = \Psi: \emptyset \times \mathbf{M} \rightarrow \emptyset. \quad [2.6]$$

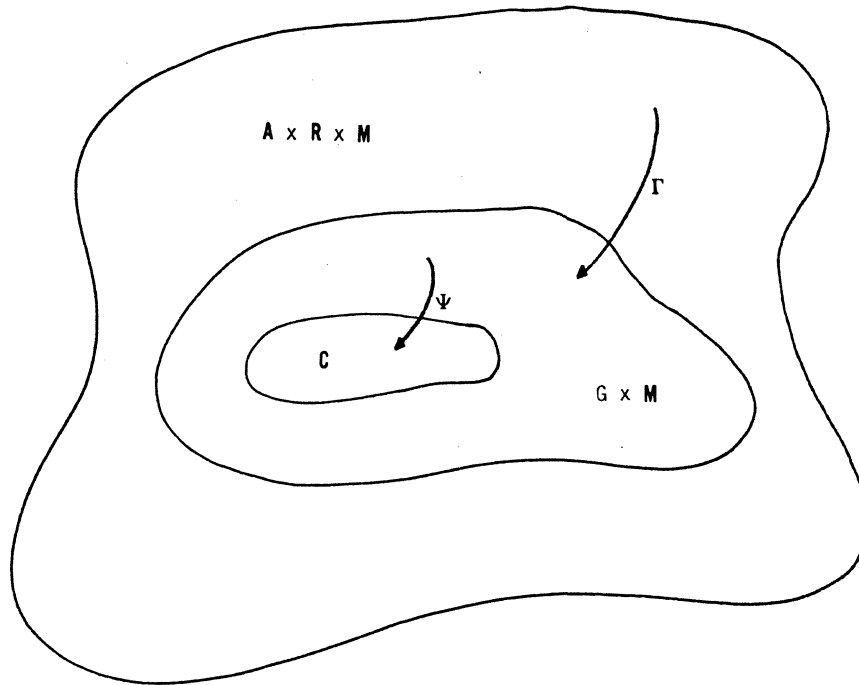


FIGURE 2.1

The Control System Design Process

The circumstance of equation 2.5 can occur if  $\underline{a}$  is not well formed (i.e., syntactically incorrect) or if  $\underline{a}$  includes a noncomputable function.

Second, if the timing constraints cannot be met,  $\Psi$  will map to  $\emptyset$ . This could be caused by function specifications requiring more time than the total monitor time available or violation of a physical law (e.g., sample rates requiring signal propagation faster than the speed of light).

Both  $\Gamma$  and  $\Psi$  are surjective since both always map to the locus,  $C$ , or  $\emptyset$ . Neither are injective since both have more than a single way to map to  $\emptyset$ , and the possibility of a single control system realizing multiple applications exists.

## REFINING THE MODEL

The Control System Design Model as expressed in the highest level of abstraction (equation 2.1) describes the process of control system production in general terms. The primary features of its approach to system development are the strict separation of application from realization, and the recognition of the necessity of temporal as well as functional congruence between application and implementation. The author is aware of the applicability of the model expressed at this high level to other areas of computing. It is an unrealized goal of computer languages to allow users to concentrate on their problems and not on the intricacies of computing solutions. It is the underlying structure of each component which enables the model to describe a specific area of system design. We are here concerned with dedicated real time control systems. Therefore, the details of each component will be defined in terms which will lead to a better understanding of the production of such systems.

### THE INTERMEDIATE LEVEL OF ABSTRACTION

Much of the difficulty encountered in producing real time control systems may be directly attributed to the duality of time frames inherent in their nature. In Chapter I, the concepts of abstract time and physical time were introduced. These two views of time will be used to describe the various properties of this time duality. Let us define physical time intuitively as a measurable period during which a physical process continues. Its measurement depends on physical phenomena, such as the rate of decay of radioactive material. Since physical time is a central parameter of descriptive models of the universe, it is in some sense an external measure. The complement of physical time is abstract time, an internal measure. It is conceptually related to "... mathematical time, [which is] of itself, and from its own nature, [and] flows equably without relation to anything external ..." described by Newton [Newton, 1729]. Abstract time is a parameter of models used to describe artifacts, the constructions of man. In the artifacts known as Petri nets for example, markers called tokens serve to measure the progress of

discrete abstract time [Misunas, 1973]. Internal processes (described by abstract time) are synthesized; they are the products of invention. External processes (dependent on physical time) are studied by analysis; their properties are discovered. If artifacts are to perform useful functions, their internal properties must be correlated with external properties of the physical world to which we are all subject. The degree of correlation required of those artifacts called computing systems can be used to functionally classify them.

Batch-oriented systems require the least correlation between their internal and external time frames. Solution by machine must be less costly than other means if it is to be selected for a particular problem. The phrase "(physical) time is money" concisely states the reason why physical and abstract time must be related at all in batch systems. The physical time spent realizing the instants of abstract time required of a program must be made as short as practicable for the sake of economy. However, the only strict requirement is that execution in abstract time be completed in some finite physical time span once initiated. The result is not dependent on the physical rapidity of the computation.

Interactive computing systems require more correlation than do batch types. System response is defined by the maximum interval of physical time consumed by primitive internal processes [Brinch Hansen, 1973]. The result of an interactive session may well vary with the degree of correlation between internal and external time frames due to effects of user attention span. The impact of abstract time (internal process speed) not meeting constraints expressed in terms of physical time (acceptable response) is on human users. While guaranteeing minimum response to users is important, an occasional unacceptably long response is not catastrophic. Physical processes are not so forgiving.

Real time systems require, by their nature, the closest correlation between abstract and physical time. When a batch or interactive system fails to meet performance standards predicated on physical time, it is said to be degraded. When a similar situation afflicts a real time system, it has failed. This difference in attitude shows that there



exists a trade-off between the importance assigned to internal parameters versus external parameters, one of which is time. The expense associated with the hardware used for batch and interactive systems has caused the internal parameters of these systems to be optimized at the expense of external ones in order to maintain a high utilization. High-cost hardware has made the system master. In real time applications, the external environment is paramount. Timing constraints must be met regardless of the degrading effects on utilization they cause. Since real time applications are the subject of this work, the correlation studied will be that of strict subordination of abstract time to physical time.

The special importance of time in real time situations makes it a natural parameter of the problem model. It is in fact the essence of the intermediate level of abstraction, just as the separation of application and realization is the essence of the highest level. A major source of the complexity encountered in producing real time control systems stems from addressing the time element as though it were of a unitary nature. As shown, this is not the case. A technique used in the physical sciences to isolate less complex forms of a time domain problem is that of variable separation [Halliday, 1950]. Such a formalism is often used to divide a problem into time-dependent and time-independent constituents, so that the properties of each part may be studied separately. The dominant aspect of the time duality in real time applications is physical time, therefore the time-dependent part is defined to encompass those facets of the problem model which must conform to physical time. It is the details of the time-dependent part which determine whether temporal congruence between application and control system is possible. The general notion of time-dependence will be represented by the symbol  $T$ . If the time-dependent part of the model is to refer to physical time, the time-independent part must refer to abstract time. The seeming contradiction of defining the time-independent part in terms of abstract time is resolved by noting that abstract time has been shown to be physical-time-independent. Abstract time is used to mark progress in the performance of a sequence of states which define a function. As such it is central to a determination of the functional congruence needed to produce a control system. The

general notion of time-independence will be represented by the symbol  $F$ , to indicate its functional (rather than abstractly temporal) nature. Sets which may be usefully divided into time-dependent and time-independent parts will be defined notationally by

$$X = (F_X, T_X)$$

where the symbol  $X$  represents such a set.

When applied to real time control systems,  $F$  represents the specification and production of a software-hardware entity capable of performing the tasks of the application (that is, capable of being mapped nontrivially by  $T$ ) without regard to external timing constraints.  $T$  supplies information so that  $\Psi$  may relate a functionally correct system to the external environment in a way which satisfies the real timing constraints of the application. The parameters of the infrastructure of  $F$  are similar to those encountered in any computational environment in which timing is not critical. However, the parameters of  $T$  are uniquely tied to the nature of real time applications. For this reason,  $T$  is expanded in the intermediate level into a triple whose components more clearly convey its character.

Real time applications consist of a group of contingencies which require responses, and a ranking of the group relating their relative criticality. Responses made to contingencies are called tasks. Contingencies sense changes in the controlled (external) environment which are significant in the context of the defining application. Such changes may become known by the controller in one of two ways: determination that a condition has been fulfilled, or an event has occurred. Likewise, the controller (by activating a task) may respond in only one of two ways: causing a condition to exist, or initiating an event. The first way provides information about the external environment that will be referred to as a condition (represented by  $C$ ), while the second provides information called an event (represented by  $E$ ). These two general forms of communication are the first two components of the triple which forms  $T$ . The third component is rank, used to order the elements of the condition and event subsets. Note that relating contingencies and their

associated tasks is correctly viewed as functional in nature and, as such, it is to be specified in the fine structure of F.

A better understanding of the general concept of "contingency" is needed if we are to examine its two constituents, condition and event. A contingency is a collection of one or more parameters ( $p$ ) of the external environment, each of which is defined by three numbers: a current value, a distinguished value, and a bound.

Those parameters which together form a particular contingency descriptor are collectively referred to as a parametric group ( $P$ ). Each contingency (and as we shall see, each task) is defined by a unique parametric group. The external environment may be viewed as consisting entirely of the parametric groups needed to specify an application.

Parameters are the fundamental units of communication and control between the physical environment and the abstract environment (the control system). At any resolvable point in time, each parameter has a value which reflects its state at that time. A contingency is said to exist for every instant of physical time that the current value of each of its parameters is equal to the corresponding distinguished value. Therefore the distinguished value of a parameter represents some significant distinguished state in the controlled environment which, in concert with other parameters, defines some distinguished occurrence we have called a contingency. Whenever one or more current values of a parametric group do not equal their distinguished values the contingency does not exist. The life span of a single occurrence of a contingency is that collection of contiguous instants of physical time during which all of its parameter values equal their distinguished values (that is, the time span in which the contingency exists). Each instance of existence will have its own life span value. The shortest possible life span in any instantial existence of a contingency (that is, the minimum possible life span) is called the bound of the contingency (symbolically written  $\beta$ ).

There exists an interesting duality between contingency and task which allows the concepts and nomenclature developed to discuss contingency to be applied to the notion of task. Contingencies define circum-

stances in the external environment which require response. Inversely, tasks define responses which must be made in answer to contingencies in order to maintain the external environment in some state that is valid in the context of the application. Therefore it is convenient to define a task to be a collection of one or more parameters ( $p$ ) of the external environment each of which is defined by three numbers: a current value, a distinguished value and a bound. Such a collection is said to be the parametric group ( $P$ ) of the task. At any resolvable point in time, each parameter (sensed by, and controlling some part of the external environment) has a current value which reflects the activity of the parameter. If at any time, all values of parameters which combine to define a task are equal to their distinguished value, that task is said to be active. It is otherwise inactive (that is, not selected by an existent contingency). The least amount of physical time in which a task must remain active in order to effect its intended change in the external environment is the bound ( $\beta$ ) of the task. To avoid confusion when discussing features which contingencies and tasks have in common, subscripts will be used in conjunction with each symbol. Contingencies are assigned the subscript one and tasks are assigned the subscript two. This reflects the nature of their relationship: the existence of a contingency causes its corresponding task to become active. It will be shown in developing the fine structure of the model that contingency and task are the fundamental concepts necessary to describe control problems.

The two aspects of contingency, condition and event, may be compared and contrasted now that the general concept of contingency has been developed. Since condition and event are but two ways of describing a contingency, it is not surprising that they have much in common. Both are defined by parametric groups. Both exist only when all current values equal corresponding distinguished values. Finally, both have a minimum life span figure (the bound). It is the duration and nature of the bound which determines whether a parametric group is an event or a condition.

Conditions are of relatively long duration (that is, they possess large bounds). Each parameter (usually more than one) of a condition definition must have this large bound property. It is useful to relate

the concept of large bound to the concept of level found in circuit theory. Assume that a contingency parametric group defines the outputs of circuits sensed by the control system. That is, each parameter of the parametric group is the output of a circuit which reflects some physical attribute of the external (controlled) environment. (It is not necessary that the model be tied to any particular method of sensing the environment, but the goal of modeling computer-based control systems makes analogies to circuit theory especially appropriate.) If these outputs are levels (continuously defined) rather than pulses (intermittently defined), then the sensed external environment may be expressed in terms of a Moore circuit model [Moore, 1956]. Portions of the environment which present this type of message to the controller are candidates for representation by conditional contingencies. Likewise, tasks whose parametric groups require levels rather than pulses to effect control are considered to be conditional in nature.

Events are of relatively short duration (that is, they have small bounds). A single parameter possessed of a small bound is sufficient to define its parametric group as an event. As with condition, event too will be related to a concept found in circuit theory: the pulse. Assume that the contingency parametric group discussed above contains one or more parameters which may be meaningfully sensed only as a pulse. Pulse outputs may be expressed in terms of the Mealy circuit model [Mealy, 1955]. The significant feature of the model in this context is the short term validity of pulsed outputs. Portions of the environment which present this type of message to the controller are candidates for event-oriented contingency specification. The duality between contingency and task holds for events as well as for conditions.

There are qualitative differences between event and condition, but they are expressible only in terms of tendencies. For example, events usually sense a single parameter, while conditions usually involve several. The dividing line is, however, quantitative: the magnitude of the bound. The bound is the maximum length of time that sensed (or applied) information is considered valid in the worst case. The final determinant, then, is whether a bound is large or small. There is no absolute answer for this in terms of the application alone. Bound

values are relative until they are specifically realized. This relativity is significant. The speed of a particular hardware/software realization of a controller is the final arbiter of whether a contingency which seems qualitatively a condition can actually be realized as one. The existence of methods for translating between Mealy and Moore circuits strengthens the argument against an absolute, realization-independent criterion for distinguishing condition from event [Hill and Peterson, 1968]. Looking ahead to the systemic interpretation of the model developed in Chapter III, events are usually assigned to interrupts while conditions are well suited to a polling approach. In the last analysis, it is the speed of the realization which determines whether interrupts are necessary for a particular contingency. This concept is stated as the principle of realization relativity: one machine's event is a faster machine's condition. The terms "event" and "condition" will be used throughout the remainder of this work to convey concepts useful in categorizing timing and environmental information, as well as concrete attributes of realizations.

The third constituent of the triple forming T describes the relative importance and interdependence which may exist between contingencies. This relative ordering is called the rank of contingencies, R. Rank is a generic term which encompasses three general concepts: priority, precedence, and global ordering of all contingencies (both events and conditions). Priority will be used to denote the level of significance of events relative to one another. This level of significance is used to determine a mode of response to simultaneously occurring events. Priority is fixed for any particular application, but may become dynamically variable in a realization which conforms to the application specification. The notion of precedence applies to contingency and task alike. A precedence relation between two entities implies an ordering. We shall consider a local time-wise order to define precedence. This may include conditional contingencies, where one contingency must be existent (or non-existent as the case may be) before another may be allowed to exist, or shared tasks, where several tasks may be initiated in some order following an enabling contingency. Global ordering simply divides all contingencies into order classes. All gradations of classes

are allowed; at one end of the spectrum each contingency might occupy a unique class, while at the other end all contingencies might be collected into a single class.

AN EXPANDED INTERMEDIATE FORM OF THE MODEL

The notions of time-independence (or functionality, F) and time-dependence (T) have been developed in some detail. Perhaps such detail would be unwarranted were these concepts not so generally applicable in refining the CSD model. It is a significant feature of the model that each of the four information sets (**A**, **R**, **M** and **C**) may be expressed in the same form at the intermediate level of abstraction. Each of these sets is divisible into a double which separates the functional from the temporal. Substitution of F and T (expanded in terms of C, E and R) into the original model definition (equation 2.1), expressed in terms of a particular application, yields:

$$CSD_a = ([F_a, (C_a, E_a, R_a)], [F_r, (C_r, E_r, R_r)]),$$

$$[F_m, (C_m, E_m, R_m)], [F_c, (C_c, E_c, R_c)], \Gamma, \Psi) \quad [2.7]$$

where

$[F_a, (C_a, E_a, R_a)]$  is a particular application specification  
(specific subset a of **A**)

$F_a$  is the time-independent (functional) specification  
of a

$(C_a, E_a, R_a)$  is a time-dependent specification of a

$C_a$  specifies condition-oriented aspects of a

$E_a$  specifies event-oriented aspects of a

$R_a$  ranks elements of  $C_a$  and  $E_a$

$[F_r, (C_r, E_r, R_r)]$	is a particular realization. (specific subset r of <b>R</b> )
$F_r$	is the internal, physical-time-independent specification of r
$(C_r, E_r, R_r)$	is the external, physical-time-dependent speci- fication of r
$C_r$	specifies the implementation of condition- oriented interactions with the external environment
$E_r$	specifies the implementation of event-oriented in- teractions with the external environment
$R_r$	specifies the implementation used to maintain the rank of $C_r$ and $E_r$
$[F_m, (C_m, E_m, R_m)]$	is a particular monitor strategy (specific subset m of <b>M</b> )
$F_m$	specifies the monitor functionally
$(C_m, E_m, R_m)$	specifies the capabilities of m to exercise time- dependent control
$C_m$	specifies the condition-oriented capabilities of m
$E_m$	specifies the event-oriented capabilities of m
$R_m$	specifies the capability of m to maintain the rank of $C_m$ and $E_m$
$[F_c, (C_c, E_c, R_c)]$	is a particular control system (specific c element of <b>C</b> )
$F_c$	specifies the controller functionally



$(C_c, E_c, R_c)$	specifies the physical-time-dependent external aspects of the controller (this collection of externally-oriented features of the controller define the controller interface)
$C_c$	specifies the condition-oriented part of the controller interface
$E_c$	specifies the event-oriented part of the controller interface
$R_c$	specifies the ranking management features of the controller interface

The expansion of the control system design model (equation 2.1) into the temporally bipartite form (equation 2.7) illustrates the basic similarity of the four information sets involved in control program production. The complexity of the model is greatly reduced by this internal consistency. Such consistency may be exploited in describing the fine structure of the model, and in relating the model to a system used to produce control systems.

#### THE INFRASTRUCTURE OF THE MODEL

The lowest level of abstraction of the model is the most detailed. This level, the infrastructure, must specify each information set of the model in sufficient detail to allow the intrinsic elements of control system design to be discerned. Therefore, each of the components of equation 2.7 will be expanded to include subcomponents which define the essential properties and functions of real time control systems. The essence of control is the recognition of significant change in the controlled environment followed by a response to the change. Significant changes are specified by contingencies and responses are specified by tasks. In order to isolate the fundamental nature of control, a generalized view of the behavior of a contingency-task pair in physical time is required.

A contingency (whether condition or event) is a description of a specific recognizable change. An agent is implied which identifies the change. Such an agent is called a discriminator. The inputs of a discriminator are the current values of the parametric group defining a contingency. A discriminator has a single binary output which is true when all current values of a parametric group equal their corresponding distinguished values (that is, when the contingency exists), and false otherwise. If the discriminator of a contingency is true, a task defined to respond to the contingency must be performed. A typical application will be composed of a number of such contingency-task pairs. A reduction in complexity accrues from viewing each pair separately, as though each one were assigned to a dedicated, independent realization. It is useful, then, to consider the features of a single, generalized contingency-task pair.

Controlling is repetitive by nature. It is a continuous process that is never completed as long as control is to be maintained. This means that a contingency-task pair describes a highly redundant course through time, like a set of beads on a string. This string of contingencies (some followed by tasks, some not, as dictated by the discriminator) spread through time defines a contingency time line. The collection of all time lines (that is all contingencies) of an application forms the time manifold of the application, and ultimately of the realization (the control system).

Consider one cycle of contingency and active task for a single time line (see Figure 2.2). Contingencies are represented by brackets and tasks by parentheses embedded in the time line. The time line is read from left to right.

There are five numbers which characterize a cycle of a time line. These numbers express three types of limiting condition:  $\delta$ , the maximum duration;  $\beta$ , the bound or deadline (previously introduced); and  $p$ , the period. Both contingencies and tasks have duration and bound values. A subscript of one indicates a contingency while two indicates a task.

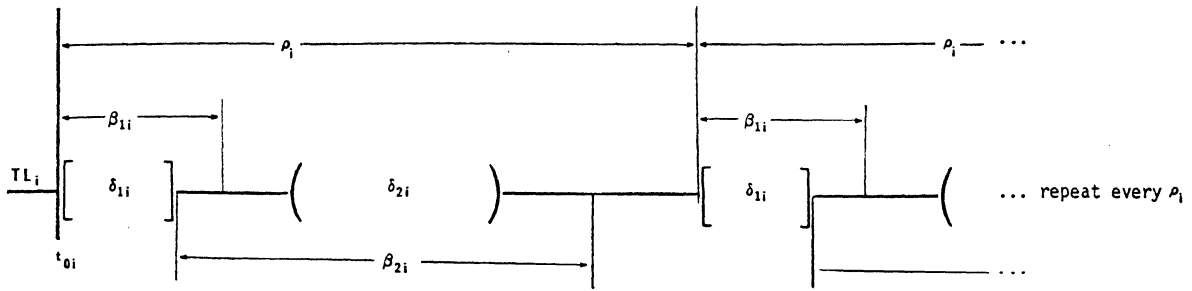


FIGURE 2.2

A Generalized Time Line

Following this convention:

$\delta_{1i}$  is the maximum amount of physical time required by the discriminator of the  $i$ -th contingency to produce its output once started

$\delta_{2i}$  is the maximum amount of physical time required to complete the  $i$ -th task once started by an instantial existence of the  $i$ -th contingency

$\beta_{1i}$  is the maximum amount of physical time allocated to the  $i$ -th discriminator to insure no lost information from the external environment.

$\beta_{2i}$  is the maximum amount of physical time allocated to the  $i$ -th task to insure no information sent to the external environment is ignored.

$\rho_i$  is the period of physical time allowed for any instance of a contingency-task pair.

The values of  $\beta$  and  $\delta$  have consistent interpretations whether they refer to condition or event oriented contingencies or tasks. This is not true of  $\rho$ , however. In the case of an event,  $\rho$  represents the minimum amount of time which can ever elapse between instances of the event. In the case of a condition,  $\rho$  represents the maximum amount of time that can be

allowed to elapse before the discriminator must be activated to avoid loss of information. The difference does not warrant separate symbols as the representation is clear in context.

These values (five for each contingency) supply the necessary global information needed to determine whether a monitor exists which can continuously schedule all contingencies to provide temporal congruence with the application. However, it is important to realize that these figures relate only to contingencies and tasks as gross entities. For example, the bound values only give deadlines for when the entire contingency or task must be completed, and say nothing of internal timing constraints. This means that even though a realization may be configured to meet all contingencies, individual operations (such as acquiring a parameter current value) may not be completed once started with sufficient rapidity to meet the demands of an application. Such time critical internal sections of contingencies and tasks are called timed blocks. There may exist several or no timed blocks in a contingency or a task. They are symbolized by  $\gamma$ , regardless of their number. The usual subscripting convention is used ( $\gamma_1$  for contingencies,  $\gamma_2$  for tasks).

The final and most detailed form of the CSD model can now be developed, since a generalized concept of timing has been devised, for timing is the basis of real time control. There are a number of subcomponents to be defined at this level of detail. Each of the elements of the intermediate level of the model (equation 2.7) will be systematically expanded to present these subcomponents. An interpretation of each element in realistic terms is found in Chapter III.

#### EXPANDED FORM OF THE APPLICATION SET

The Application Set may be expressed in terms of its functional and temporal parts:

$$A = (F_A, (C_A, E_A, R_A)). \quad [2.8]$$

Expanding each component of **A**, we have:

$$F_A = (P_1, P_2) \quad [2.9]$$

$$C_A = (\beta_1, \beta_2, \gamma_1, \gamma_2, \rho) \quad [2.10]$$

$$E_A = (\beta_1, \beta_2, \gamma_1, \gamma_2, \rho) \quad [2.11]$$

$$R_A = (\pi, \eta) \quad [2.12]$$

where

$P_1$  represents a contingency parametric group of **A**

$P_2$  represents a task parametric group of **A**

$\beta_1$  is the bound of  $P_1$

$\beta_2$  is the bound of  $P_2$

$\gamma_1$  is the internal timing (timed block) bound of  $P_1$

$\gamma_2$  is the internal timing (timed block) bound of  $P_2$

$\rho$  is the period of  $P_1$

$\pi$  is the local order (priority or precedence) of  $P_1$

$\eta$  is the global order of  $P_1$ .

The pair  $(P_1, P_2)$  represents the parametric groups of a contingency-task pair, and therefore defines functionally the time line for a contingency. The remaining parameters (equations 2.10 through 2.11) specify those timing and rank values which must accompany a functional definition of an application if it is to be complete.

## EXPANDED FORM OF THE REALIZATION SET

The Realization Set may be expressed in terms of its functional and temporal parts:

$$\mathbf{R} = (F_R, (C_R, E_R, R_R)) \quad [2.13]$$

Expanding each component of  $\mathbf{R}$ , we have:

$$F_R = (H, \Delta_H, \Lambda_H, T_H) \quad [2.14]$$

$$C_R = (S_C, (I_C, \lambda_C)) \quad [2.15]$$

$$E_R = (S_E, (I_E, \lambda_E)) \quad [2.16]$$

$$R_R = (S_R, (I_R, \lambda_R)) \quad [2.17]$$

where

$H$  is a hardware description of a central processing unit  
(or units)

$\Delta_H$  is the Algorithm Translator Mapping of  $H$

$T_H$  is the Timing Mapping for the hardware ( $H$ )

$\Lambda_H$  is the Interface Hardware Selection Mapping of  $H$

$S_C$  is the condition-oriented interface software

$(I_C, \lambda_C)$  is the condition-oriented interface hardware

$I_C$  describes condition-oriented interface hardware

$\lambda_C$  is the latency of  $I_C$

$S_E$  is event-oriented interface software

$(I_C, \lambda_C)$  is event-oriented interface hardware

$I_C$  describes event-oriented interface hardware

$\lambda_C$  is the latency of  $I_C$

$S_R$  is ranking software

$(I_R, \lambda_R)$  is ranking hardware

$I_R$  describes ranking hardware

$\lambda_R$  is the latency of  $I_R$ .

The functional component of the set  $\mathbf{R}$  is formed of an information set and three mappings. The set describes a particular central processor. The transformations allow the hardware to be related to software (programs) and to physical time. The  $\Delta_H$  mapping defines a process of relating algorithm to a form executable by H. For example, if S is an unrealized algorithm description and  $S^H$  is the same algorithm in a form useable by processor H, the S and  $S^H$  are related by

$$S^H = \Delta_H S. \quad [2.18]$$

$S^H$  is said to be the H-dependent form of the algorithm S.

The  $\tau_H$  mapping determines the physical time required to execute an algorithm (expressed in H-dependent form). If we assume that parametric groups may be expressed algorithmically,  $\Delta_H$  and  $\tau_H$  provide a means of determining a value for  $\delta$ , the maximum (physical) duration, for a particular processor H:

$$\delta = \tau_H \{ \max[\Delta_H(P^H, S^H)] \} + \lambda^H \quad [2.19]$$

where max is a function determining the maximum execution paths through both the algorithm representing the parametric group and the ancillary interface programs.  $\lambda^H$  represents a composite of the latency figures associated with the interface programs ( $\lambda_C^H, \lambda_E^H$  and  $\lambda_R^H$ ).

Those subsections of parametric groups called timed blocks may be examined separately (for comparison to their internal bounds,  $\gamma$ ). The run time for a timed block is represented by  $\sigma$ , and given by:

$$\sigma^H = \tau_H \{ \max[\Delta_H(\bar{P}^H, \bar{S}^H)] \} + \lambda^H \quad [2.20]$$

where  $\bar{P}$  and  $\bar{S}$  are sections of P and S related to the timed block.

The mapping  $\Lambda_H$  defines criteria by which interface hardware is selected to relate abstract and physical time. It is, in a sense, a hardware counterpart of  $\Delta_H$ . Therefore, unrealized interface hardware needs are selected specifically for H by  $\Lambda_H$ :

$$I_C^H = \Lambda_H I_C. \quad [2.21]$$

Here again the superscript H indicates an H-dependent format.

While the functional component contains elements associated with abstract time, the temporal component specifies those portions of the realization which must perform duties bounded by physical time. The hardware and programs which maintain the relation between the processor (H) proceeding in abstract time and the external environment form the interface of the realization.

#### EXPANDED FORM OF THE MONITOR SET

The Monitor Set may be expressed in terms of its functional and temporal parts:

$$\mathbf{M} = (F_M, (C_M, E_M, R_M)) \quad [2.22]$$

Expanding each component of  $\mathbf{M}$ , we have:

$$F_M = (P_M) \quad [2.23]$$

$$C_M = (D_C, O_C) \quad [2.24]$$



$$E_M = (D_E, O_E) \quad [2.25]$$

$$R_M = (D_R, O_R) \quad [2.26]$$

where

$P_M$  is the parametric group functionally defining a monitor ( $m$ )

$D_C$  specifies the condition-oriented operational definition of  $P_M$

$O_C$  is the condition overhead portion of  $P_M$

$D_E$  specifies the event-oriented operational definition of  $P_M$

$O_E$  is the event overhead portion of  $P_M$

$D_R$  specifies the rank management operational definition of  $P_M$

$O_R$  is the rank management overhead portion of  $P_M$ .

$P_M$  is a parametric group functionally defining a monitor. Monitor parametric groups have as their parameters contingency discriminator outputs (on which initiation of tasks is based), and task initiators. A task initiator is a parameter which causes a task to be started when its current value equals its distinguished value. This means that all parameters of a monitor parametric group reflect the status of entities residing in the internal environment (a discriminator) or the interface (such as a time-slice clock). This contrasts with contingency and task parametric groups whose parameters define activities only in the external environment. This division of environmental scope between **A** and **M** is necessary if complete realization-independence of **A** is to be maintained. The application must refer only to the external environment. While there is no necessity for restricting monitor parameters to interface and internal considerations, such a restriction does permit sufficient information for monitor definition. Further, a clean separation is more easily understood and allows a modular approach to realizations.

The temporal aspects of **M** are specified in two parts: the operational definition and the overhead. The operational definition specifies the capabilities of a particular monitor. These capabilities are usually but not exclusively expressed in terms of relations between various values defining application time lines (such as  $\beta$ ,  $\delta$ , and  $\rho$ ).  $D_E$  for instance, defines the operational capacity of a monitor to respond to events. A value of  $\emptyset$  indicates no event processing is available.

Overhead defines the cost in realization resources required by the monitor to maintain control.  $O_E$  defines both the general overhead consumed by any event and specific overhead consumed by specific forms of event. An example of overhead is the time ( $\omega$ ) used by the monitor. The maximum time overhead for event processing, for example, is given by

$$\omega_E^H = \tau_H \{ \max[\Delta_H(O_E)] \} + \lambda^H. \quad [2.27]$$

Note that  $\omega_E^H$  is defined relative to some realization  $H$ .  $\omega_E$  cannot be given a realistic physical time value until it is realized.

#### EXPANDED FORM OF THE CONTROL SYSTEM SET

The Control System Set may be expressed in terms of its functional and temporal parts:

$$\mathbf{C} = (F_C, (C_C, E_C, R_C)) \quad [2.28]$$

Expanding each component of **C**, we have:

$$F_C = (H, (P_1^H, P_2^H, P_M^H)) \quad [2.29]$$

$$C_C = (S_C^H, I_C^H) \quad [2.30]$$

$$E_C = (S_E^H, I_E^H) \quad [2.31]$$

$$R_C = (S_R^H, I_R^H) \quad [2.32]$$

where

$H$  is a hardware description of a central processing unit

$(P_1^H, P_2^H, P_M^H)$  is software/firmware in a form compatible with  $H$

$P_1^H$  is a contingency parametric group of an application in  $H$ -compatible form

$P_2^H$  is a task parametric group of an application in  $H$ -compatible form

$P_M^H$  is a monitor parametric group in  $H$ -compatible form which satisfies temporal congruence of an application

$S_C^H$  is condition-oriented interface software in  $H$ -compatible form

$I_C^H$  is  $H$  compatible condition-oriented auxiliary hardware

$S_E^H$  is event-oriented interface software in  $H$ -compatible form

$I_E^H$  is  $H$  compatible event-oriented auxiliary hardware

$S_R^H$  is ranking interface software in  $H$ -compatible form

$I_R^H$  is  $H$  compatible ranking auxiliary hardware.

The functional component of a control system consists of central processing hardware ( $H$ ) and software. (Software is used generically and therefore includes alternate realizations such as firmware.) The software encompasses both contingency and task ( $P_1^H$  and  $P_2^H$ ) needed to functionally realize the application, and the monitor software ( $P_M^H$ ). Monitor software is included in the functional constituent of  $C$  since it operates in abstract time. An interface with the external environment (proceeding in physical time) is needed to complete the control system and bring it into subservience to physical constraints. Such an interface is specified by the temporal part of  $C$ .

The temporal part includes both software and hardware needed to temporally interface the functionally correct control system internal environment to the external world. Each of the three subparts of temporal congruity (condition, event, and rank) may require both software ( $S^H$ ) and interface hardware ( $I^H$ ). The temporal part of **C** is known collectively as the interface of the control system, while the functional part is known as the controller. Together they form a control system.

#### USE OF THE MODEL

The CSD Model, even at the most detailed level of abstraction, maintains a stylized perspective. This allows it to remain useful through future computer generations. It is likely that the major technological elements of the model will be used for the foreseeable future: the hardware-software dichotomy and the need for an interface. Further, the abstraction of the controlled process and the controller as entities bound to physical and abstract time frames which require coordination reveals a fundamental attribute (and source of difficulty) in machine control. However, we should like to apply the insights gained from the model to the specific task of dedicated real time control. Chapter III interprets the model in light of current technology.

#### REFERENCES

- M. A. Aizerman, Theory of Automatic Control, Pergamon Press, London (1963).
- J. F. Barkley, Jr., "Does Interactive Computing Happen in Real-time?," SIGMINI Newsletter, ACM, Vol. 1, No. 3 (November 1975) pp. 2-3.
- M. Biewer, Microprocessors for Dedicated Control, Pro-Log Corporation, Monterey, CA (1974).

- R. Boudarel, J. Delmas, and P. Guichet, Dynamic Programming and Its Application to Optimal Control, Academic Press, New York (1971).
- D. M. Bowers, "Data Acquisition and Logging," Modern Data (September 1975) pp. 55-60.
- P. Brinch Hansen, Operating System Principles, Prentice-Hall, Englewood Cliffs, NJ (1972).
- J. A. Cadzow and H. R. Martens, Discrete-time and Computer Control Systems, Prentice-Hall, Englewood Cliffs, NJ (1970).
- E. G. Coffman, Jr. and P. J. Denning, Operating Systems Theory, Prentice-Hall, Englewood Cliffs, NJ (1973).
- E. W. Dijkstra, "GOTO Statement Considered Harmful," CACM, Vol. 11, No. 3 (March 1968) pp. 147-148.
- E. W. Dijkstra, "The Structure of the 'THE' Multiprogramming System," CACM, Vol. 11, No. 5 (May 1968) pp. 341-346.
- Albert Einstein, Relativity, the Special and General Theory, Henry Holt and Company, New York (1921).
- Mark Elson, Concepts of Programming Languages, Science Research Associates, Chicago (1973).
- E. R. Fisher, "Micro versus Mini in a Laboratory Environment," Proceedings of the CUBE Symposium, Lawrence Livermore Laboratory, Livermore, CA (October 1974).
- S. H. Fuller, D. P. Siewiorek and R. J. Swan, "The Design of a Multi-Micro-Computer System," Proceedings of the Third Symposium, SIGARCH News, Vol. 4, No. 4 (January 1976) p. 123.
- H. R. J. Grosch, "High Speed Arithmetic--The Digital Computer as a Research Tool," J. Optical Society of America, Vol. 43, No. 4, (April 1953) pp. 306-310.
- David Halliday, Nuclear Physics, John Wiley, New York (1950).
- H. Hermes and J. P. La Salle, Functional Analysis and Time Optimal Control, Academic Press, New York (1969).

- I. N. Herstein, Topics in Algebra, Blaisdell Publishing, New York (1964).
- F. J. Hill and G. R. Peterson, Switching Theory and Logical Design, John Wiley, New York (1968) pp. 295-298.
- G. A. Korn, "A Survey of Minicomputer Applications," Comput. and Elect. Engng., Vol. 2, Pergamon Press (1975) pp. 3-33.
- D. Layzer, "The Arrow of Time," Scientific American (December 1975) pp. 56-69
- J. A. N. Lee, The Anatomy of a Compiler, Reinhold, New York (1967).
- T. H. Lee, G. E. Adams and W. M. Gaines, Computer Process Control: Modeling and Optimization, John Wiley, New York (1968).
- C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," JACM, Vol. 20, No. 1 (January 1973) pp. 46-61.
- W. C. Lynch, "Operating System Performance," CACM, Vol. 15, No. 7 (July 1972) pp. 579-585.
- G. K. Manacher, "Production and Stabilization of Real-Time Task Schedules," JACM, Vol. 14, No. 3 (July 1967) pp. 439-465.
- M. N. Matelan, "Parallelism in Automatic Testing," Proceedings of the 1975 Sagamore Computer Conference on Parallel Processing, (August 1975) pp. 100-104. Also available as technical report UCRL-77132 from the Lawrence Livermore Laboratory, University of California, Livermore, CA.
- M. N. Matelan, "Automating the Design of Microprocessor-based Real Time Control Systems," ACM-IEEE Design Automation Conference Proceedings, (To be published 1976).
- G. H. Mealy, "A Method for Synthesizing Sequential Circuits," Bell System Tech. J., 34, No. 5 (September 1955) pp. 1045-1080.
- D. Misunas, "Petri Nets and Speed Independent Design," CACM, Vol. 16, No. 8 (August 1973).

E. F. Moore, "Gedanken Experiments on Sequential Machines," in C. E. Shannon and J. McCarthy (eds.), Automata Studies, Princeton University Press, Princeton, NJ (1956).

Issac Newton, Mathematical Principles of Natural Philosophy and His System of the World, translated by Andrew Motte (1729), revised and annotated by Florian Cajori, Univ. of California Press, Berkeley, CA (1947), Vol. 2, p. 6.

A. Opler, "Requirement for Real-Time Languages," CACM, Vol. 9, No. 3 (March 1966) pp. 196-198.

B. Randell, "Towards a Methodology of Computing System Design," Proceedings of the Nato Science Committee on Software Engineering, Scientific Affairs Division, NATO, Brussels (October 1968) pp. 204-208.

K. Rothmuller, "Task Partitioning in Programmable Logic Systems," Computer, Vol. 9, No. 1 (January 1976) pp. 19-24.

J. D. Schoeffler, "The Development of Process Control Software," Proceedings of the Spring Joint Computer Conference (1972) pp. 907-914.

H. A. Simon, "The Architecture of Complexity," Proc. American Philosophical Society, 106, 6, (1962) pp. 468-82.

Omri Serlin, "Scheduling of Time Critical Processes," Proceedings of Spring Joint Computer Conference (1972) pp. 925-932.

Henry David Thoreau, Walden, Bramhall House, New York (1951 Edition).

A. Tucker, "Very High-level Language Design: A Viewpoint," Computer Languages, Vol. 1, Pergamon Press (1975) pp. 3-16.

# OFFICE OF NAVAL RESEARCH

BRANCH  
OFFICE  
LONDON  
ENGLAND

## ONR LONDON REPORT

R-13-75

COMPUTER AIDED DESIGN AND DESIGN AUTOMATION  
IN EUROPE

WALDO G. MAGNUSON, JR.

20 OCTOBER 1975

UNITED STATES OF AMERICA

This document is issued primarily for the information of U.S. Government scientific personnel and contractors. It is not considered part of the scientific literature and should not be cited as such.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

### V. CAD AND DA ON THE EUROPEAN CONTINENT

#### A. Summary

The development of CAD and DA in Italy and Germany compares closely with that in Britain. Germany, although lagging behind the US in computer utilization in engineering design, appeared to have better and more numerous contacts with US research in CAD and DA than either Italy or Britain has. Philips in Holland is taking a very aggressive approach to developing company coordination in CAD. Similarly, Siemens in Germany has a sizable group in CAD. Olivetti in Italy over the years, has developed extensive DA systems quite independent of other research. Among the universities, the University of Karlsruhe, Technische Hochschule Darmstadt and University of Bologna all have active and excellent research programs in CAD and DA.

#### B. European Short Reports

##### Germany

Technische Hochschule Darmstadt, Darmstadt (Prof. R. Piloty)

Considerable work in the development and implementation of a register transfer language for digital design automation.

University of Karlsruhe, Karlsruhe (Prof. R. Hortenstein)

Register transfer level language development and symbolic and block diagram notation for digital design are active research areas.

Computer Gesellschaft Konstanz, Konstanz (Dr. H. Schwarzer)

Extensive DA development for designing digital computers. Work includes RT and gate simulation, wiring, test pattern generation, construction support, all tied to a common data base written in COBOL.

Siemens Central Laboratory for Data Techniques, Munich (Drs. B. Rehn & D. Essl)

Considerable work in network analysis, optimization, transistor modeling, digital simulation and test generation, PCB layout and artwork generation, and programs for MOS chip design and layout.

Technische Universität Berlin, Berlin (Dr. R. Hoffman)

High level digital description language development (RTL) and program development for microprogram and flip-flop realization.

Munich Technical University, Munich (Prof. Saal)

Fundamental investigation in network synthesis and analysis. CAD program development in circuit optimization.

University of Saarlandes, Saarbrücken (Dr. M. Glesner)

Network analysis program development with emphasis on linear and nonlinear component sensitivity and nonlinear network optimization using both matrix-based and symbolic-based analysis programs.



## Italy

- Politecnico di Milano, Milan, Italy (Prof. V. Anoaia)  
CAD research in analysis of large electrical networks. Optical pivot techniques, sparse matrix techniques, tearing of networks are all being studied. Past work in dc, sensitivity statistical, ac analysis of linear networks.
- University of Bologna, Bologna (Prof. V. Monoco)  
CAD design of microwave circuits with considerable work in tolerance analysis.
- Marconi Italiana, Genova (Mr. G. DeLotto)  
Linear network analysis program development. Synthesis and approximation of fitters, narrow band crystal filter design, and optimization of filter designs are all being done by computer programs.
- Olivetti S.p.A., Ivrea (Mr. E. Majorana)  
Very impressive array of DA programs for PCB design including logic schematic drawing, design rule checking, partitioning, placement, etc.
- Siemens S.p.A., Milan (Dr. A. Illa)  
CAD efforts consist of electronic circuit analysis, filter analysis, and digital filter design. DA efforts are aimed at PCB design, wire-wrap routing, and documentation logic simulation with fault diagnosis; and test generation is also an active project.
- Telettra, Vimerate (Mr. F. Molo)  
Considerable work in computer program development for synthesis and analysis of filters for FDM equipment. DA work in automatic diagnosis of logic circuits, automatic generation of masks for PCB, wirewrap drive tapes, and recent work in logic simulation and hardware documentation.

## The Netherlands

- Philips-Nijmegen (Mr. J.G.M. Klomp)  
AC and transient analysis, MOS cell modeling, logic simulation, test generation, MOS cell placement and interconnection, and NC tape production all aimed at CAD for MOS chips.

## C. Europe CAD and DA Reports

### Design Automation at Olivetti

Approximately thirty-five miles south of the Matterhorn the small town of Ivrea, Italy, stands at the mouth of the Valle d'Aosta. The large Olivetti typewriter and calculating machine factory is located there and accounts for about two-thirds of Ivrea's near 35,000 population.

Olivetti electronic computer-aided design (CAD) started in 1964 with the work by Ing. Enzo Majorani who is now director of CAD efforts. From 1964 when the first PAO (Progettazione Automatica Olivetti or design automation) program ran on an Olivetti ELEA 9003 computer until today, when there are about 300 PAO programs operational on an IBM 370/158 computer, a very impressive complement of computer aids have been developed. The design automation (DA) group has grown from three people in 1964, to 25 in 1970, to the present 30. The composition of the current group is two engineers, 13 programmers (IBM trained, high school diploma level) and 15 key-punch and computer operators.

The PAO design system goals throughout the development have been: standardization, normalization, documentation, turn-around, design power, flexibility, man-system interaction, modularity, and reliability. Majorani believes all have been reached; and after viewing input procedures, documentation, and production results, I am inclined to agree with him fully. Olivetti's PAO system is one of the most polished systems I have seen, not only in results but in completeness.

The aim of the PAO system is to produce documentation (signal lists, component placement lists, and logic schematics) and numerical control (NC) drive tapes (artwork plotter and drilling machine) using Olivetti standard and special printed-circuit boards (PCB) starting from a logic designer's rough logic schematic. This procedure is not so unusual or different from DA systems used elsewhere. What does make the PAO system outstanding is the extent to which they have developed many of the procedures in going from input to output.

Olivetti uses a 16 by 22 inch drafting sheet with an 8 by 9 grid for data entry. The designer, from working logic schematics, positions a module in each rectangle. Ten input and output signals are allowed on each module, although a module can be continued onto as many rectangles as necessary. Each input/output signal is given a name, thus allowing signal strings to be constructed by the computer system. The information on this work sheet becomes the input source for keypunching. On computer processing, a logic master file is created which becomes the basis for later design changes, computer processing, and record retention.

An early output of the DA system is plotter (AEG plotter) produced logic schematics. For example, the designer needs only to draw any block or symbol to represent the logic function, and the computer program will produce MILSTD-806B drawings with complete signal routing and a signal list. The position of the modules are in the same relative position as input by the designer. Careful attention has been given to the routing so that interconnect lines do not pass through signal names modules, or module titles. I was very impressed with the layout, legibility, and appearance of the drawings.

The real payoff at Olivetti, however, is in the automatic printed-circuit board (PCB) production. Their standard boards (7 by 10 inches) accept ninety 14 or 16 pin modules. Last year they completed approximately 500 two-sided PCB's and this year will layout about 650. From the logic master file, the design is first partitioned (by program) to boards. These boards are then optimally placed in a back panel. The back panels are themselves printed circuit boards, and in placing the plug-in cards, not only are the cards interchanged but the signals on the pins are interchanged so as to align interconnection paths. An automatic routing algorithm then completes the back-panel boards after which the appropriate NC tapes are generated for producing the boards. Alternatively, the interconnection can be done manually using a digitizer.

After the completion of the back panels, Olivetti has two methods of designing the individual PCB. The fully automated method is usually employed, but there is an alternative manual allocation and routing which can be used. In the manual mode the normal method is to use rough pencil layouts followed by digitizing. The manual mode data is fed into the logic master file. The interesting part, however, is the computer mode and the degree to which they have automatized operations. As previously mentioned, before the back-panel boards are designed, the logic is automatically partitioned from the schematic onto standard cards. This operation presents few problems, according to Majorani, because they use only two board sizes and generally are dealing with modest size systems.

The allocation program performs several operations. The first function is to assign the logic, which has been partitioned to the board, to integrated circuit packages. It is not necessary for the designer to do this. The next program function is to place the packages on the board. The placement algorithm is a constructive technique in which additional weight is put on connections to the edge connector. After an initial placement, an optimizer routine is used which, pair-wise, first interchanges logic elements and then integrated circuit packages. Finally, an optimization procedure is used to order the signals on the terminal pads of each module (for example, interchanging two inputs on the same gate).

After the allocation is completed, a routing program is called by the system. The steps in routing taken by their program includes: wire-list determination, layering (only two-side PCB's are dealt with), ordering of the wire lengths for each side, an initial wire layout, minimization of corners and vias, modification of wire layout, and finally making a list of connections which were not completed. For the uncompleted connections, they use a digitizer and manually route the paths.

Following a final check of the connection paths, the NC data for the artwork plotter and drilling machine is generated. An additional important part of the output is the documentation. A components insertion diagram is produced on the plotter as is an updated logic schematic. There is also documentation output for automatic testing and simulation, although these two items do not play a terribly important role in their DA system.

The Olivetti PAO system has been evolving for over ten years and today is an extremely important part of their design and production process at Ivrea. According to Majorani they have developed their procedures with very little contact or exchange of information with other companies. I was particularly impressed with the completeness of their system. For example, their automatic partitioning to boards and of logic elements to integrated circuit packages is exceptionally effective and their system is one of the few in actual day-to-day use. I was also impressed with the quality of the logic schematics which their PA system produced.

Majorani admitted that special printed-circuit boards continue to be a problem. Generally, these are accommodated by manual intervention at the back-panel design level for interconnection and for allocation and routing at the PCB level. Manual operations are done by digitizer with feedback to the logic master file. Future work will incorporate into the system more automation for special PCB's and possibly the extension of the entire system to handle multi-layer printed-circuit boards.

To date the Olivetti PAO system has not been described in the open literature. Certainly they have one of the better systems and it deserved recognition.

#### CAD ACTIVITY AT SOCIETA ITALIANA TELECOMUNICAZIONI SIEMENS S.p.A.

Siemens S.p.A. (S.p.A. = Societa per Azioni = share company) is located near Milan and employs approximately 30,000. The corporate structure has three divisions: Production, Marketing and Sales, and Engineering Research and Development. The Research and Development Division has 1600 people and my host for the visit was Dr. A. Braga Illa who is Director of Technical Coordination. The group in the R & D division under Dr. Mario Bonatti is responsible for most of the computer aided engineering - design efforts. Thirty-four engineers are engaged in three areas of developing computer programs for use in engineering.

The computer aided design (CAD) group is involved in developing programs for system analysis and functional simulation of electrical networks. They also write programs for system analysis of transmission systems. A recent product of this group is a program named SICL for logic circuit simulation which is still under development and is rather modest in the size of networks it can accommodate. It has a limit of 400 signals but does have built-in macros for a wide range of elements (flip-flops, multiplexers, encoders, arithmetic logic units, random-access memories, etc.). The delays may be specified for individual gates. As further work is done on the simulator, they hope to incorporate fault detection and test sequence generation as part of the system.

The computer aided engineering (CAE) unit is the largest (17 people) of Bonattis' groups engaged in applying computers to engineering problems. They are active in printed-circuit board (PCB) layout, the physical design of back-panel wiring, and the generation of wire-wrap control tapes. The CAE system developed at Siemens S.p.A. makes use of Univac 1106 computer and a drafting machine made by the Quest Company. The input is in terms of the network topology and module physical descriptions. This input is checked for errors and consistency, after which one of two programs is used. If the circuit consists of micromodules (dual inline packages), the elementary logic functions (gates) are grouped and assigned to distinct modules. The assignment is made according to a scheme which strives to minimize the interconnections between modules, that is, it maximizes internal module connections and the pin-to-pin connections on each individual module. Next, the modules are placed on the board in an ordered manner. Starting from this initial placement, the modules are then interchanged based on the objective function of reducing the total wire length by using the connection matrix values. Strategies are employed to reduce the number of module exchanges and to recognize when it is no longer advantageous to interchange modules.

If the board components are prevalently discrete components, a force-placement technique is used. The assumptions are that an attraction force exists between interconnected points proportional to their separation and that a repulsion force inversely proportional to distance exists between all components. The algorithm attempts to find a lowest total energy potential of the system. This method is patterned after the scheme first described by Fisk et al in Proc. IEEE 55 No. 11 (Nov. 1967). After assignment and placement, a routing program is employed.

The routing program first establishes a priority for the connections to be made. The algorithm separates the tracks into vertically and horizontally preferred segments, and then, using feedthroughs and a channeling routing method, the routing is completed. Usually, greater than 98% board completion is achieved for 230 x 253 mm boards containing 55 modules. Approximately 10,000 PCB's are designed each year. Besides the 10 people involved in the operation of the automated layout system, over 100 are doing manual layout, and about 10 are using a digitizer for layout.

The third group is engaged in computer usage for engineering design for both hardware and software for system analysis for telecommunications systems.

In looking to the future, Bonatti stated that they will be extending the printed circuit automation in the direction of multilayer boards. They will be continuing their development of a logic simulation and test diagnosis program, and they hope to develop a more extensive library of design components. Finally, they hope to move in the direction of interactive computer graphics, particularly for placement and routing.

#### TELETTRA'S ACTIVITY IN COMPUTER AIDED DESIGN

Telettra was founded in 1946 by a small engineering group devoted to the design and manufacture of telecommunication equipment. Today, it is one of the principal telecommunication concerns in Italy and is the only privately owned one in that country (Fiat recently acquired a 35% share). The company has grown to about 4500 people and approximately 500 are electrical engineers. The main offices and factory are located at Vimercate, 12.5 miles northeast of Milan, and a new (1970) branch factory for production is located in Trieste.

The organizational structure of the company follows rather conventional lines with a Planning Group, a Research and Development Department, a Quality Control Department, a Production Department, and a commercial organization. The Production Department is essentially divided into five broad branches which include equipment for multiplex telephone systems, radio-relay systems, data transmission systems, transmitters and repeaters for broadcast, and traffic control systems. The contact for my visit was Francesco Molo, who is manager of research and Development. After giving an introduction to company organization and products, Molo turned me over to Giuseppe Dallargine, director of Communications, and Giuseppe Stacchiotti, head of the filter and crystal laboratory.

The computer aided design (CAD) activities at Telettra are centered in two subject areas. One is in the FDM (frequency division multiplexing) Transmission System Division where Stacchiotti leads development of computer aids for the synthesis and analysis of filters. CAD activities in this area have been in progress since 1960 with several computer programs, based on rather conventional filter design approaches, having been developed. More recent work, since 1971, has been in the area of developing programs for assisting in digital filter design. All filters today at Telettra are designed and analyzed using computer aids.

To me, the more interesting CAD work is by a small group (about seven) in design automation (DA) under the direction of Dallargine. They have been active for approximately four years with the objective of developing automatic or semi-automatic processes for assisting in the design and production of digital telecommunication equipment. The work so far has been focused on the automatic diagnosis of logic circuits, automatic generation of masks for printed-circuit-boards, and generation of the drive tapes for semi-automatic wire-wrap wiring machines. Recently, emphasis has been placed on logic simulation and automatic generation of documentation for the hardware projects.

The automatic diagnosis software programs that have been developed have centered about purchased test machines (General Radio) for testing rather modest-sized printed-circuit logic boards. Full fault coverage and diagnosis were objectives in designing the software. In their work on automatic routing for printed-circuit-boards, the research has been not so much towards developing new algorithms but of testing the various types of software techniques available on the market. The results of their studies were unavailable, and it was my impression that the bulk of the printed circuit layout is being accomplished by hand.

One DA item in use is the computer generation of control tapes for semi-automatic wire-wrap machines used in wiring the sub-racks for printed-circuit-boards. Starting from a library containing the connector descriptions for all the printed-circuit-board types, they generate the necessary wire-wrap control tapes.

Still in planning and investigation stages are software aids for logic simulation and project documentation. The software for logic simulation that has been developed to date is based on the Ulrich algorithms (E. G. Ulrich, "Time-sequenced logical simulation based on circuit delay of active network paths," Proc. ACM 20th Nat. Conf., 1965). At present, the programs are being run on a Univac 1106 computer, but their use in interactive simulation using minicomputers is being investigated. Telettra is also studying the possibility of developing programs for drawing schematics and component layout drawings, but as yet little progress has been made along these lines.

Although the use of computer aided design and design automation has not reached wide spread use, management appears to be well aware of the potential payoffs. The group engaged in DA is young, enthusiastic, and just getting started. We can look forward, I think, to a much more rapid development and commitment to CAD and DA design aids in the near future at Telettra.

#### RESEARCH IN CIRCUIT THEORY AND DESIGN AT THE UNIVERSITY OF BOLOGNA

Bologna, Italy, is a city noted for having the oldest university in Europe. The University (Universita degli Studi di Bologna) was founded in 425 and had 10,000 students in the 13th century. Today the enrollment is over 50,000.

My visit in Bologna was to the Institute of Electronics at the University where my congenial host was Prof. Vito A. Monaco. Monaco and his colleagues have been engaged in developing computer programs for the analysis of electrical circuits and involved in computer-aided design since 1968. One of the early products was a computer program (SCAWAT) for circuit analysis from dc to microwave using scattering parameters.<sup>1</sup> This program has been used since 1969 by the students at the University in their classes in electronics. In addition, the program has been used at universities in Palermo, Switzerland and Denmark and in industry at Telettra and Thomson-CSF.

Past research work has included projects involving automatic scattering matrix methods applied to computation of microwave amplifier circuits. Their work has also included active device modeling by lumped-element equivalent circuits, by means of measured port parameters, and in terms of geometrical dimensions and the electrical characteristics of the materials forming the components (as for microstrip transmission lines).

More recent research has been in the areas of sensitivity analysis of linear networks, optimal design procedures for component value spread and network function tolerances<sup>2</sup>; sparse-matrix techniques for linear networks<sup>3</sup>; computer-aided microwave circuit analysis and multivariable optimal design of electronic circuits.<sup>4</sup> One product of this research has been an analysis program BMT using sparse-matrix techniques with optimum equation, ordering and pivoting.<sup>5</sup> The program can supply various network functions such as wave vectors, circuit response functions, first-order sensitivities, and group delay.

The group has ample computational facilities available in the form of a CDC-6600 system, a HP 2001 and 2116B, and a little-used connection to an IBM 360 system. The 6600 was joined in January by two CDC Cyber 70's (Models 76-12 and 72-14). The three Control Data machines will provide computer resources for 12 universities and technical institutes.

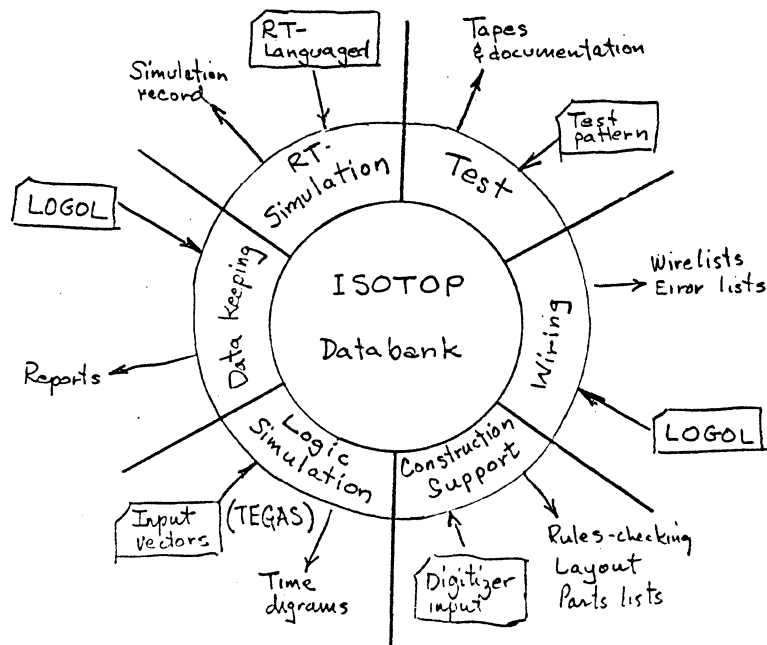
A majority of the research is supported by the Italian Research Council (CNR). Monaco outlined a current proposal to CNR for 18 million Lire (\$27,000) for half-time support of ten students and faculty for further work in sparse-matrix microwave analysis, microwave amplifier studies, and tolerance analysis. The research by the group has been tied to quite practical problem areas (particularly in the area of microwave circuits) due to Monaco's enthusiastic leadership and because of his and his associates' close relationships with industry. The group is strong and the future looks bright for computer-aided circuit theory and design at Bologna.

Although the future existence of the computer-aided design (CAD) group at Computer Gesellschaft Konstanz (CGK); Constance, Germany is currently in doubt, the group is fairly large (approximately 20) and has developed several rather powerful CAD systems. The question of the future stems from the July 1974 acquisition of the company by Siemens AG. Prior to last year the company was a subsidiary of Telefunken Computers who had concentrated its computer design at Constance after takeover in 1958. Since the recent takeover by Siemens, the development of the next computer range has stopped, and it is highly probable that the CAD activity will move to Siemens-Munich.

The CAD activities began in 1962 with the definition of a language (LOGAL) for describing digit logic systems. LOGAL was coupled with a sequential magnetic-tape-based data base called LOTSE. The two systems have evolved through many stages to their present-day status.

LOGAL, as it exists today, is a free-format, block-structured language. Each logic function on a printed-circuit board (PCB) is described by the type of function, name of function, technical title of function and the signal-pin assignments on each module. The symbolic placement of each module on a PCB can also be supplied as input. The LOGAL description serves as input to the new data-base (called ISOTOP) for either wiring purposes or for data storage purposes.

The replacement for the LOGAL-data-base system was a new system called RUE (Rechnerunterstützten Entwicklung = Computer-aided design). The RUE system has several portions (see figure): The input language (LOGAL), a library, the databank (ISOTOP), a data storage operation procedure, a logic simulator, a construction support part, a wiring portion, a test generation part and a register transfer simulation part.



RUE - System for CAD and Construction

These parts are all interface programs with the RUE ISOTOP data base, a direct access storage system (disk-based) written in FORTRAN and COBOL. The data base system (DBS) is a subset of the COBOL language which was chosen as the implementation language because of its data management facilities. The structure of DBS is oriented towards PCB's, and variations in particular PCB designs, which may occur in different computers in a series, are accommodated. They have implemented several methods for storage and retrieval including direct, sequential, index-sequential, near-by and hashing methods for data storage and retrieval. Early in a design, a

private file system is used by the designer. When the design reaches the point where there can be compatibility problems, it is then transferred to the public file system and all users interact with each other. The DBS system is operational on a Telefunken TR-440 computer system and uses 25 M bytes of storage. The system has been operational for 1-1/2 years.

A. Hoyer, the leader of the DBS development, stated that the most difficult part was to specify the requirements for the system. They consider both a block and a net oriented organization. Most CAD data bases are organized with construction in mind (routing PCB's, parts lists, wire lists, etc.) and are net oriented. But people who begin with simulation prefer a block-structured data base. When changes in a design must be conveniently accommodated, a net or signal oriented data base may be difficult to manipulate and a block structure is better. Hoyer said that because they had to cover the range from design to manufacturing to the field phase as well as take into account later versions of a digital system, they decided to use a block structured design. The actual realization in COBOL was not so different, however. Altogether, approximately 35 man-years were invested in DBS. Hoyer mentioned that to the best of his knowledge, their data base development was the first (reported) direct-access CAD data base.

A good deal of the CAD development was motivated initially by the design of the TR-440 computer. The TR-440 is a network micro-programmed computer. No micro-programs are stored in micro-storage but rather are stored in the topology. Therefore there are many different PCB's in the computer, many of which are used only once. This led to work in developing a test generation program for testing PCB's. The Armstrong procedure was programmed and used for generating test patterns. Because the TR-440 cards have a pin/package ratio smaller than 0.5, the program was not very successful and many patterns had to be developed by hand. The next development was to implement a Boolean difference technique which has been very successful.

In developing their test sequence generation program, they did not include a general logic simulation program. About a year and one half ago they purchased the TEGAS program for gate level simulation. Hoyer feels that logic simulation is worthwhile only at higher levels, over at least a full mother-board for PCB's and generally simulation of a full processor. They were interested in simulating systems containing about 70K gates (10 mother-boards each composed of 40 PCB's with each PCB having about 200 modules). The difficulty with simulating smaller blocks of logic is the interface problem (inter-connecting of signals where the topology is partitioned). In fact, the number of interface errors is so large when simulating at the PCB level, it isn't worthwhile. At Constance they are modifying the TEGAS simulator to be able to simulate 100K gate networks. They do not plan on using the test generation part of TEGAS.

To date, they have used no facility for automatic routing of PCB's and all boards have been hand designed. The rules for the new boards they are designing are very difficult (MECL logic utilizing two impedances on multi-layer PCB's) and nearly impossible to check manually. A rules-checking program was written which makes the appropriate signal checks for the multilayer boards. Because CGK was unable to use an earlier developed routing program based on Lee's algorithm for the two-impedance multilayer boards, they have been developing a placement and routing program. The placement uses the Steinberg assignment method and the routing is by means of a two-step procedure using first, channel routing and then detailed (step) routing. The philosophy of the program uses a combination inter-linking the placement and routing phases. The program has achieved 98% path completion for the channel portion, and for the remaining 2% the step routing completed 95%.

As mentioned earlier, the future for computer design (and thus for CAD development) is very unsure at CGK. The best surmise is that future design automation work will be done at Siemens in Munich. Whether the CGK data base system which has focused on the problems of computer development (logic simulation, PCB testing and design rules checking) will be able to be successfully joined with the Siemens PENTA system which has concentrated on fabrication (layout, routing and wiring) is something only time will tell.

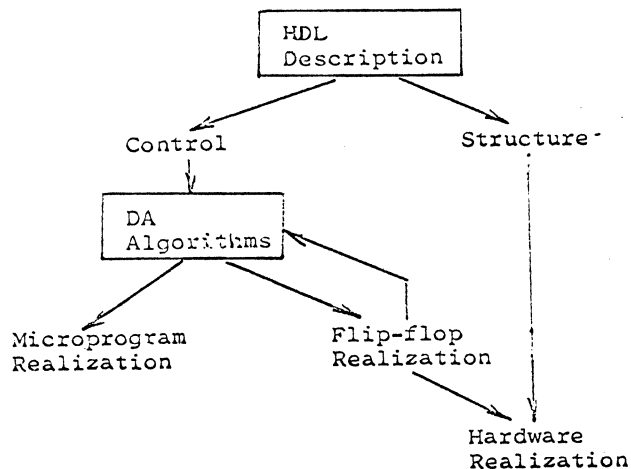
The programming of digital computers has advanced from absolute binary and machine code in the late 1940's and early 1950's through assembly language to the present-day emphasis on machine-independent language programming (e.g. the use of high-level languages such as FORTRAN, ALGOL, PL/I, APL). In a similar fashion the level of design of digital systems (computers) has changed with time. Early systems were designed at a circuit level in which logic gates were described by interconnections of diodes, transistors, resistors, etc. (and vacuum tubes and relays). As solid-state technology advanced, design advanced from a component to a switching circuit level where the logic is expressed by a combination of gates and flip-flops. While components became smaller, the complexity of functions contained on IC chips grew. In designing digital systems with these functions, engineers have become increasingly aware that methods of abstraction similar to those available in programming (sub-routines, macros, optimizers) would be very beneficial and are often necessary. To quote Prof. Yaohan Chu, University of Maryland (*Computer*, Dec. 74): "Would a mathematics instructor be willing to teach a course on algebra or differential equations without the use of symbols? Probably not. ... Likewise, computer-hardware description languages, which are nothing but another application of symbology, are essential to computer design."

Dr. Rolf Hoffmann of the Institute for Technical Informatics, Technical University Berlin, has recently defined a hardware description and programming language (HDL). For the past five years he has been part of a research effort involving general topics in computer organization and logic design. This work, led by Prof. H. Liebig, has centered on four areas: design automation of switching circuits, computer design hardware languages, memory organization, and computer structures.

HDL, introduced by Hoffmann, can be used as: a matrix programming, a hardware description and a microprogramming language. The language was mainly defined to describe the different levels of hardware, i.e., algorithmic, functional behavior and the hardware level for all timing details. Asynchronous systems, clocked sequential circuits and general digital systems can be equally well described by the language. HDL also combines properties such as matrix operators, the definition of subunits (MACROS) and subroutines and special operators for asynchronous circuits. HDL was mainly influenced, Hoffmann stated, by the languages APL, CASSANDRE, CDL and ISP (see *Computer*, Dec. 74). From the language specification, the next task will be to develop a compiler and simulator.

In addition to the above work, the group has also been developing a complete design automation (DA) system starting from the HDL description of a computer. They have written PL/I language programs to describe the control part of the description and APL programs for the decomposition, minimization, covering problem, state reduction, and substitution property partitioning. The structure part has not yet been specified (see figure Page 50).

The PL/I programs translate the description to a switching table for the network. This table is then processed and a flip-flop realization is computed. They also intend to compute a microprogram realization; but when the state table is transformed to a read-only-memory (ROM) microprogram, it is not possible to accommodate all of the inputs - the ROM would be too large. They are working on a procedure to reduce the number of inputs.



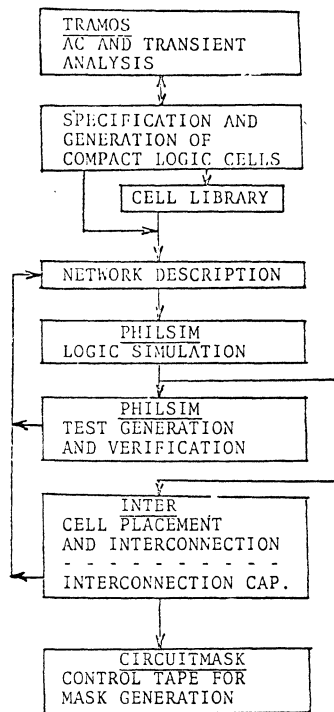


APL will probably be the target language for the realization of HDL and for the implementation of a DA system. Hoffmann believes APL is well suited to logic design problems and is particularly suited for coding switching functions for simulation purposes.

#### COMPUTER AIDED DESIGN AT PHILIPS-NIJMEGEN

The Integrated Circuits Development Laboratories of N.V. Philips' Gloeilampenfabriken is located in Nijmegen, Netherlands, which is one of eight main Philips' factories in the Netherlands and located about 32 miles NE of the company headquarters in Eindhoven. I visited Ir. J. G. M. Klomp who is in charge of the computer aided design (CAD) efforts at Nijmegen. Klomp's group, about ten strong, is principally concerned with integrated circuit (IC) development and in particular MOS technology. About 35 MOS designs have been completed during the year that the Nijmegen CAD system has been operational.

The figure represents the flow of different phases in a design through the Philips CAD system. Computer program names are underlined.



The operation of the CAD system allows the flexibility of either full design from specification to MOS integrated circuit masks by the CAD design group or design by the customer with only guidance by the CAD group. In either case, the starting point in regard to the software aids is the library of IC cells. The library contains about 100 building blocks ranging from simple gate patterns to rather complex patterns like register units and even Boolean function patterns. The shapes of all cells are of constant height and variable length.

An important feature for all cell patterns is that all input and output signals are brought out on both the upper and lower sides of the cell. The advantage of bringing patterns out on both sides of the cell is that the interconnect routing problem is drastically simplified because it is rarely necessary to route around a cell. Also the problem of twisted power supply lines does not occur because a cell has been placed upside down to gain interconnection area. A set of basic patterns is also available for designing specific cells. The patterns are on separate diffusion levels based on the technology. Their use results in cells that fit into the layout system and that do not conflict with the technology constraints.

By use of the basic patterns and newly designed cells, the transistors, capacitors, diodes, resistors and inputs are specified for the TRAMOS program. This program calculates the electrical behavior with respect to time. In this way, the circuit response can be checked and the basic building blocks being designed can be verified to be compatible with those from the library.

The function of TRAMOS is to check the basic building blocks and how they function together but not to check the complete network. The network is checked with the logic simulator program PHILSIM. This is an event-driven two-level simulator with provisions for specifying delays and fan-in/fan-out loading. PHILSIM shares a lot in common with Fairchild Semiconductor's FAIRSIM logic simulator. Klomp has had close association with Fairchild and early in developing Philips' CAD system a not fully satisfying experience was realized in trying to get FAIRSIM to operate on a Philips IBM 360 computer. The program PHILSIM is able to simulate networks with several thousand gates on the medium-size Philips computer at Nijmegen. It can also start with the simulation control statements and verify a test pattern to isolate single-error faults. Each test pattern is input by the user, and the program enables him to see whether his logic can be tested and how efficiently it is done.

The philosophy of the placement routine is one of simplicity, that is, only a first-order placement is attempted. The main effort has been put into making the inter-connection program very efficient. The efficiency is in large part due to the double rail connection terminals in the basic cells. After interconnection, the designer inspects the layout (via a graphics console) and then decides if an exchange of cells can save chip territory. Densities of 90-100 gate-functions per  $\text{mm}^2$  are achieved in this way.

Finally a program, CIRCUITMASK, processes the information built-up in the above interactive process and generates the necessary code to drive a pattern generator or plotter.

In developing their CAD system, Klomp's group has paid particular attention to the cost-effectiveness in the use of computer aids. They find that savings are possible in the areas of total turn-around time, error-free transfer of the design data from one phase to another and production of documentation. For example, a circuit of about  $7 \text{ mm}^2$ , which took a designer nine months to complete by hand, was later re-designed in nine weeks with the CAD system and used the same chip area.

In developing CAD software, the effort has moved in the direction of leaving the creative part of the design to the designer rather than the computer. This has enabled them to write smaller, faster and more efficient programs. Also they have particularly aimed at using small computers, again for reasons of efficiency in overall computer use. Almost all programming is in FORTRAN.

Klomp and his group have developed a smoothly running CAD system that appears to be very cost-effective. He states that all LSI mask generations are done on the system, and in fact, they would be unable to do them by hand. Towards future developments and improvements, they are working on more refined test pattern generation and verification techniques and are further extending their simulation program in terms of speed and fidelity.

## VI. CONCLUSIONS

While visiting universities and companies in Europe I have often been asked "how does CAD or DA here compare with that in the US?" There is no one answer. I found places where the research and practice of CAD and DA is very good and on a par with that in the US. Generally, however, I found the level of DA practice and commitment (at comparable sized companies) much lower than that in the US. Usually engineering access to computing power is substantially behind that in the US. Interactive systems are considerably less evident in both universities and in research laboratories. Often I found small companies using practices which had been discarded several years ago in the US for economic reasons. As stated earlier, it is my judgement that in CAD and DA, Europe is behind the US by up to five years. In no place did I see evidence where it surpassed US technology.

PROCEEDINGS FOR 9th, 10th and 11th DA WORKSHOP

The following is the rate schedule as agreed to by SIGDA and the DA Technical Committee of IEEE.

COPIES	SIGDA & ACM Members	All Others	Amount
1) _____ 9th DA Workshop Proceedings (1972) @ \$10.00 376 Pages	\$10.00	\$16.00	_____
2) _____ 10th DA Workshop Proceedings (1973) @ \$10.00 288 Pages	\$10.00	\$16.00	_____
3) _____ 11th DA Workshop Proceedings (1974) 379 Pages	\$12.00	\$20.00	_____
Member Number _____		TOTAL =	_____

Please send your order prepaid to:

ACM Inc.  
P.O. Box 12105  
Church Street Station  
New York, New York 10249

Make your checks payable to Association for Computing Machinery  
(an added charge is made for billing).

---

JOIN SIGDA    JOIN SIGDA    JOIN SIGDA    JOIN SIGDA    JOIN SIGDA    JOIN SIGDA    JOIN SIGDA    JOIN SIGDA

\_\_\_\_\_  
Name (please type or print)

Annual membership dues  
are \$3.00 for ACM members  
and \$5.00 for others.

\_\_\_\_\_  
Affiliation

\_\_\_\_\_ Enclosed annual dues.

\_\_\_\_\_  
Mailing Address

\_\_\_\_\_ Please send more info.

\_\_\_\_\_  
City                      State                      Zip

Mail to SIGDA  
ACM INC.  
P.O. Box 12105  
Church Street Station  
New York, NY 10249

# Association for Computing Machinery

1133 Avenue of the Americas, New York, New York 10036

Non-Profit Org.  
**U. S. POSTAGE**  
**PAID**  
New York, N. Y.  
Permit No. 2397