SPECIFICATION, SIMULATION AND AUTOMATED DESIGN OF INTERFACES AND DIGITAL CIRCUITS

by

Alice Parker

DRC-18-5-79

January 1979

Department of Electrical Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213

Army Contract #DAAG29-76-G-0024

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

SECURITY CLASSIFICATION OF THIS PAGE *(Whon Dmtm Enttod)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

A, TITLE *(and Subtitlo)*

Specification, Simulation and Automated Design of Interfaces and Digital Circuits

5. TYPE OF REPORT & PERIOO COVERED

Final; *; *£ayv>l, 1976- May 30, 1978

6. PERFORMING ORG. REPORT NUMBER

7. AUTHORO)

Alice C. Parker

8. CONTRACT OR GRANT NUM8ERO)

DAAG29-76-G-0024

9. PERFORMING ORGANIZATION NAME ANO AOORESS

Alice C. Parker
Electrical Engineering Dept., -Carnegie-Mellon Univ.
Pittsburgh, PA 15213

10, PROGRAM Ei.EMEN1, PROJECT, TASK AREA 4 WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME ANO AOORESS

U.S. Army Research Office
Post Office Box 12211
Research Triangle Park, NC 27709

12. REPORT DATE

July 31, 1978

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & AOORESSff/ *dltimrmnl from Controlling OtilGO)*

15. SECURITY CLASS, *(oi thtm roport)*

Unclassified

15«. OECL ASS! FtCATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(oi thim Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(oi thm mbotrmct ontmrod in Block 20, it dittmrmnt from Rwport)*

NA

16. SUPPLEMENTARY NOTES

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS *(Continum on rormrmo midm it nocmmmmty and idtMtty by block numbor)*

I/O, Interface, Design Automation, Simulation, Logic Design, Hardware Descriptive Language, Bus Specification, Computer-Aided Design.

2O. ABSTRACT fC«t£au« « wf *tdooittr *r block number)*

This report describes research done in hardware description, simulation, and design automation. Although the basic thrust of the work has been aimed at I/O and interface problems, most of the results are more general.

The efforts in formal hardware description have produced a language for bus, I/O and interface specification, GLIDE. GLIDE is supported by a compiler which performs syntactic and semantic checks. A translator to the ISPL language has also been written. The resulting code and

DD FOR* 1473 EDITION OF 1 NOV SS IS OBSOLETE
\ JAM 73

Unclassified
SECURITY CLASSIFICATION OF T*.S PAGE

20. (cont.)

non-translatable GLIDE semantics forced abandonment of ISFL either directly or indirectly from GLIDE for I/O description. An example of GLIDE describing the UNIBUS*[31] is included here.

In order to simulate I/O and bus transactions, changes were made to the ISP simulator to allow specification of timing and independent PROCESSES. A further experiment in simulation was done with ISP descriptions of micro-code execution for a dozen commercial processors.

The major research reported here is design automation work, part of a larger design automation project at Carnegie-Mellon University. This grant has supported synthesis research - the mapping from a functional description of the system (ISP) to be designed to the structure. Automated design of the control circuitry is in the early stages, but a working program designs the data paths, registers and memories. An early design the program produced came within 25Z of the cost of the design a human designer produced. The PDP-8/E design, included here, has a chip count within 50% of that commercial design, but the design program produced an entirely different design.

TABLE OF CONTENTS

1.0 SCOPE OF THE INVESTIGATION

The motivation behind the research described in this report is to enhance the digital designer's capabilities by producing more powerful design tools. Digital logic design has progressed to the point that the operation of the logic can be functionally expressed by a variety of hardware descriptive languages, ISP being one of the more widely used ones. Functional simulators exist and are useful for verifying system operation and performance measurements (Barb77a). Thus, the state of the art in digital design is such that the next addition to design aids should be synthesis, a program that can design the STRUCTURE of a digital system, given its FUNCTION or BEHAVIOR as input. Along with the synthesis of hardware comes the problem of producing optimal or near optimal designs to meet the design constraints. The research reported on here is aimed at understanding the design or synthesis process so that it can be automated. One of the goals of this project is to produce logic-level hardware designs from ISP descriptions in a non-optimal fashion to better understand automated design. (A parallel goal of related research by the same group is to develop discrete optimization algorithms and technique to be applied in a more complex and powerful package.

Unfortunately, the area of I/O interface and bus design is not as well organized or developed as digital design. In order to produce design aids for this kind of problem, much more background effort has been necessary. First of all, specification of bus and I/O operation is a different, more difficult problem than logic description.

Second, simulation is more complex due to timing dependencies which affect the logical operation of the interfaces to I/O and buses• So, in order to automate interface design, the remaining effort on this grant has been aimed at the problem of I/O interface and bus description. The goal here has been to produce a language suitable for I/O description and simulation, with the automation of complex interface design a more distant goal. At the same time, the synthesis programs described above have been constructed so that reasonable, simple interfaces to the hardware being designed can be specified and included in the design.

In the course of pursuing the above goals, seme other areas have been investigated. These include the specification of a module set data base for interface designs, the comparison of the interface specification language (GLIDE) with ISPL, and the description and simulation of microcode execution for a number of processors. In the area of I/O design, some further specifications of a general-purpose programmable I/O processor were produced, and a programmable FIFO buffer chip design was investigated. Publications and technical reports in these areas are listed in Section 3«

2.0 SUMMARY OF RESULTS

We are presenting here two main research results - the GLIDE language, and a working synthesis program, the data-memory allocator. Related conclusions and results are also briefly enumerated.

2.1 The GLIDE Language

A summary of the GLIDE language progress is presented here. Since the complete language has not been published elsewhere, a pre-publication report is attached as Appendix I. The GLIDE language has now been completely specified, and a compiler supports the language. Early effort went into the comparison of GLIDE and ISPL, and this work produced a compiler which translated GLIDE into ISPL. The result of this was a better understanding of the limitations of ISPL, and the introduction of the PROCESS concept in the ISPS language and simulator. By PROCESS we mean the set of register-transfer operations which exist in a control environment independent from the control environment of other operations. In addition, timing capabilities were added to the ISPS simulator. Some of the GLIDE control structures could not be translated accurately into ISPL, and some primitive GLIDE operations expanded into large blocks of ISPL code. In particular, the GLIDE memory constructs include FIFO queues and associative memories, which expand into long routines when translated to ISPL. Also, the semantics of GLIDE contain the notion of synchronous data I/O, which cannot be described in ISPL. Other Primitive operations which translate to routines include parity bit generation and checking, data formatting, and packing and unpacking of

words. The major conclusion to be drawn from the comparison is that GLIDE and ISP are different languages for describing different entities, and that the problems with I/O description force the existance of both languages - GLIDE for I/O and ISP for digital systems.

The maor output of the GLIDE effort so far is two partial bus descriptions - the military computer GYK/12 I/O bus and the PDP-11 UNIBUS™. The UNIBUS description is attached as Appendix II. Three main conclusions can be drawn from these two examples. First, the control structures for nesting PROCESSES have some undefined semantics, and it is not obvious the effect the PROCESS priority structure should have on the execution of a GLIDE program. Second, the control structures inside processes are not block structured, and hence unwieldy. However, the descriptions seem to accurately reflect the logical operation of the two buses, and therefore, the language is viable for bus and I/O description. (Attempts to describe the UNIBUS with ISPL and ISPS have not resulted in complete descriptions). Efforts are underway to validate the GLIDE UNIBUS description.

2.2 Design Automation

In order to discuss the results of the design automation effort, an overview of the RT-CAD (Register-Transfer level Computer-Aided-Design) system is presnted here. This overview was originally published in (Snow78a).

RT-CAD OVERVIEW

The ulmate goal of -the RT-CAD project is to provide a technology-relative, structured-design aid to help the hardware designer explore a larger number of possible design implementations. Inputs to the system are a behavioral description of the system to be designed, an objective function which specifies the user's optimization criteria, and a library specifying the hardware components available to the design system. The components of the RT-CAD system are shown in Figure 1 and discussed below.

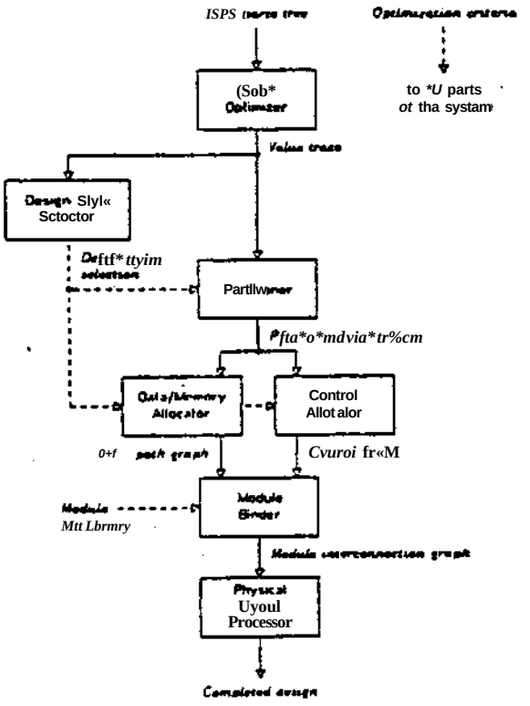

figure I: fK-CAO System Overview

The RT-CAD system differs from other design automation systems in that it operates from a behavioral specification. Such specification provides a model that, while accurately characterizing the input-output behavior of a piece of hardware, does not necessarily reflect its internal structure. The design process is one of binding implementation decisions in a top-down manner as a design proceeds through the RT-CAD system. More and more structural detail is frozen at each level until a complete hardware specification is obtained, the most influential design search space. The functions of the design system components which bind these implementation decisions are described below.

GLOBAL OPTIMIZER. The global optimizer applies high-level transformations to a design's behavioral representation after translating it from ISPS notation (Barb77b) to an abstract design representation called the value trace (Snow78b). The transformations have a significant impact on the cost, performance, and other parameters of the designs to which they are applied. The research described in this paper centers around the design representation, the transformations upon it, and the strategy guiding their application in the search for an optimal implementaion.

DESIGN STYLE SELECTOR. By considering the various module sets that can be used (e.g., TTL vs. a microprocessor), the design constraints imposed (e.g., cost, speed), and the structure of the algorithm to be designed (e.g., pipeline data flow), the design style selector decides on the specific style of design to be employed (e.g.,

bitislice microprocessor, MOS microprocessor, SSI/MSI logic). Earlier work (Thom77b) shows this to be an influential decision in terms of cost and speed tradeoffs. When the style is selected, the design is passed to an allocator specific to the design style. Initial research into the design style selection process has been completed (Thorn 77a) and an automatic design style selector is currently being programmed.

PARTITIQNER. The partitioner groups operations from the abstract design representation into control steps. This effectively binds the control flow for the design. Tradeoffs between the data and control parts are made at this level.

DATA/MEMORY (DM) ALLOCATOR. The function of the DM allocators is to decide the number and type of data operators, muliplexors, and registers needed to implement the data part of the design. They are style specific in that they embody analytic and heuristic knowledge about a style (e.g., the trade-offs involved in the design of a TTL system), but they do not have access to the specific details of each module set. The output of the allocator is a data path graph whose nodes are elements such as adders or registers. An initial implementation of an allocator for the TTL design style is reported in (Hafe78).

CONTROL ALLOCATOR. The control allocator generates a sequential state machine to control the data paths produced by the DM allocator. The control allocator has the option of designing the control unit around control philosophies such as microprogramming, programmed logic arrays, random logic, etc. The output of the control allocator is a

control path graph whose nodes represent control states.

MODULE BINDER. The module binder selects physical modules from the module set library to implement a design's data and control path graphs. The library contains descriptions of the components available to the design system and may be freely updated so that it is kept current with respect to advances in module technology. This dynamic aspect of the module set library provides for the technology-relative aspects of the RT-CAD system.

PHYSICAL LAYOUT PROCESSOR. This component partitions the system into printed circuit boards or chips, decides the placement of components, routesinterconnections, and prepares engineering documentation.

Research is currently underway into the design of all of the system components described above. In addition, the problem of integrating them into a coherent design system is being investigated.

Research supported under this grant has focused on the synthesis routines - the data-memory and control allocators. Although the control allocation effort is just beginning, some ideas as to the nature of the problems to be solved have been posed. The generation of control hardware is analogous to the problem of generation of microcode, with its inherent computational complexity, but there is one difference. Generation of hardware introduces another set of variables into the optimization routines. Not only are microinstructions generated, but the control hardware itself must be

designed and optimized.

More progress has been made on the data memory allocation problem. A non-optimizing allocator has been written and reported in (Park78) and (Hafe78). In order not to duplicate these publications, (Hafe78) is attached as Appendix III, and the results are summarized here. This allocator produces a distributed logic design of the data paths and storage locations for a given ISPL description). (The program uses ISPL instead of ISPS because of the ISPS development timetable. It is being modified to accept ISPS). It performs some error checking to indicate to the user potential resource conflicts and design errors, and functions independently of the actual integrated circuits used to implement the logic diagram it produces. Preliminary checks indicate that the designs are capable of performing the functions present in the original description. Two designs have been done by the allocator. The first is part of an elevator controller and is described in Appendix III. The second is the PDP-8/E. A non-optimal hand mapping of integrated circuits onto the allocator output logic diagram has been done, and estimate of chip count made. It is difficult to compare the automated design with the original DEC design for three reasons. First, the ISPL description input to the allocator declares as registers some values the PDP-8E uses but never stores explicitly in registers, such as the effective address. These show up as registers in the allocator's design. Also, the allocator designs distributed logic, and the DEC design was done in the central-accumulator design style (For a discussion of design styles, see (Thom77)). Finally, the DEC design has assumed a boundary

between the control and data-memory parts of the design, but the boundary is different from that imposed' on the allocator by the ISPL description. Thus some tests, flags, and registers which must be declared explicitly in the ISPL description are part of the control in the DEC design. In spite of these differences, estimates of chip count indicate that the allocator uses 50% more integrated circuit chips than the human designers for the data paths and registers. Of course, these estimates were made using the same 1970 technology chip set the DEC designers had to deal with. The 50% excess hardware can be found in muliplexers which connect the registers, the extra registers declared in the ISPL description, and duplicated operators like increment, add, and compare. Much of this can be attributed to the way in which the ISPL description had to be written, and some of these constraints will not be present in future ISPS descriptions. However, other chips can only be eliminated when optimization algorithms operate at some stage of the design process. The complete allocator output can be found in Appendix IV, along with the implementation information used to make the chip count estimates, and the PDP-8/E ISPL description.

One interesting point to be illustrated is the differences in the design seen even from the block diagram level. This is shown in Figure 2. There are two reasons for the differences. First, as stated previously, the design styles are different. Second, the multiplexing is used in different ways. In the DEC version, the operators are shared, and are even used to provide no-op paths from one register to another. In the CMU version, only registers are

shared and use multiplexed inputs. The ISPL language is partially the source of this disparity. In ISPL, the user can repeatedly use register A as a destination from various sources• However, the expressions A+B and C+D do not imply (or discount) a single adder. Other differences in the design include the use of multiplexers for shifting in the DEC design, and use of true/complement 0/1 chips for creating complements. "Oring" of the MQ and AC registers in the DEC version is done within the multiplexing hardware. Constants are often created in one place and gated over already existant data paths to the registers. In the CMU version, these constants are multiplexed at the register inputs.

One final difference is the treatment of the Link FF and Accumulator register as a single register in the CMU version. This is done because of the way the PDP-8/E ISPL description was written. Further analysis of this design is in progress and includes an implmentation of the control by hand. Comparisons of the DEC and CMU speeds will then be possible.

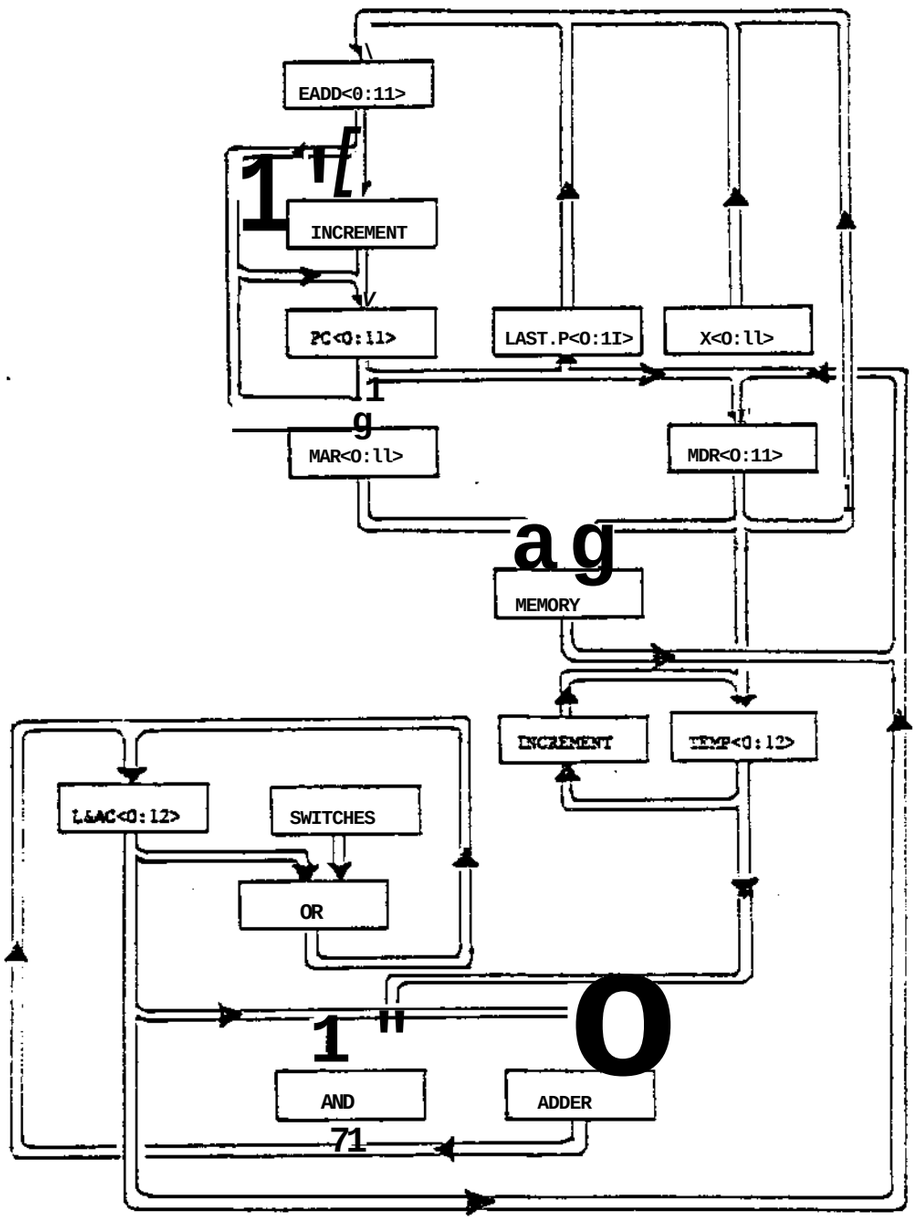

Figure 2a. Sleek Biagraa of *031* ?D?-3 Design.

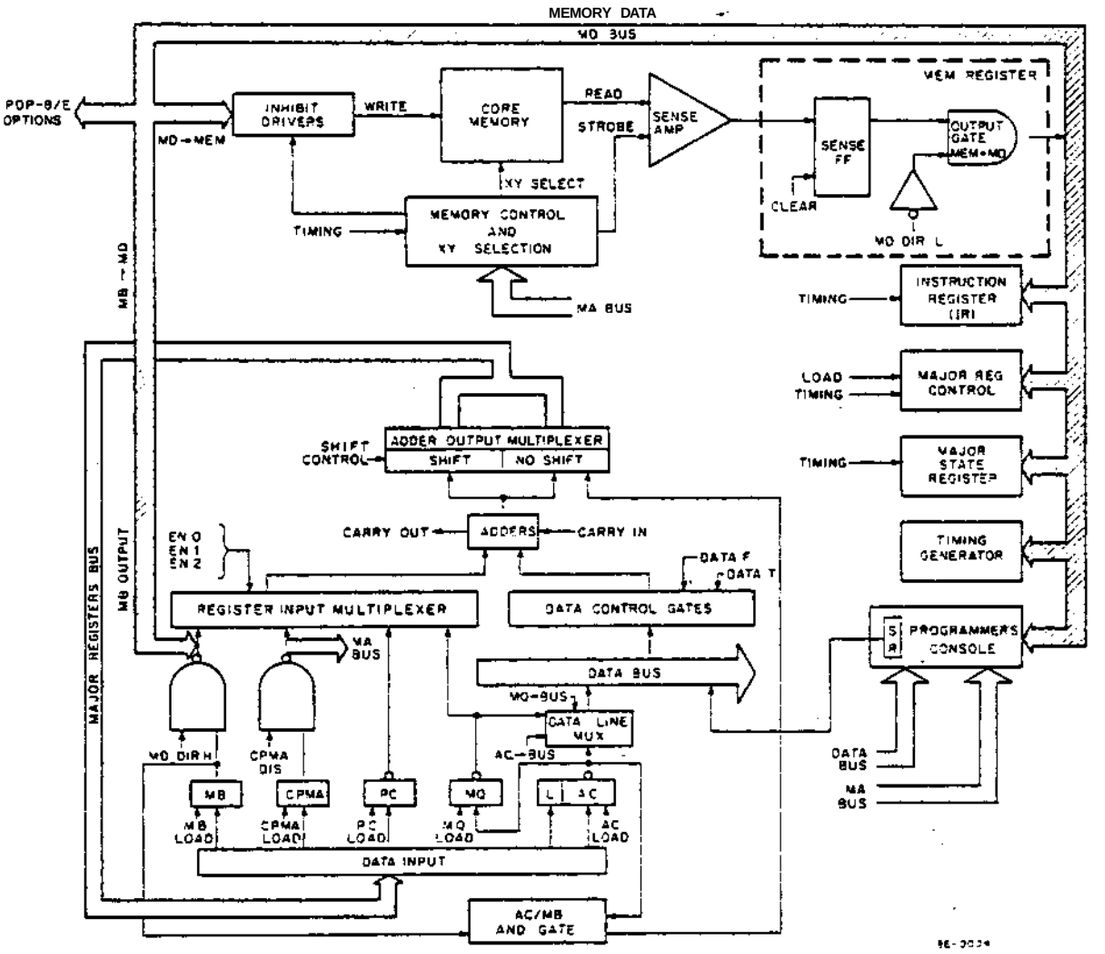

Figure 2b. Basic Data Paths,

3.0 PUBLICATIONS

"An Investigation of Glide - A Generalized Language for Interface Description and Evaluation," Andrew Nagle, M.S. Project Report, Carnegie-Mellon Universty, Electrical Engineering Department, August, 1976.

"Hardware/Software Tradeoffs in a Variable Word Width, Variable Queue Length Buffer Memory,[11] A.C. Parker with A.W. Nagle,. Proceedings of the 4th Annual Computer Architecture Symposium, March, 1977.

"Register Transfer Level Digital Design Automation: The Allocation Process," Louis Hafer and Alice Parker, Proceedings of the 15th Annual Design Automation Conference, June, 1978.

"The Application of a Hardware Descriptive Language for Design Automation," Alice Parker and Louis Hafer, Proceedings of the Third Jerusalem Conference on Information Technology, August, 1978.

"Data-Memory Allocation in the Distributed Logic Design Style," Louis Hafer, M.S. Project Report, Carnegie-Mellon Universty, Electrical Engineering Department, December, 1977.

"Automatic Design of Sequencers for the Control of Digital Hardware," Andrew Nagle, Thesis Proposal, Carnegie-Mellon University, Electrical Engineering Department, January, 1978.

"The Development of a Hardware Descriptive Language for Interfacing," [1] Alice Parker, Andrew Nagle, and Bill Lyden, Carnegie-Mellon University, Electrical Engineering Department Technical Report, August, 1977.

"Digital Interface Description," Alice Parker, Proceedings, COMPCON, February, 1978.

"Description and Simulation of Microcode Execution," Alice Parker and Andrew Nagle, Proceedings of the 5th Annual Symposium on Computer Architecture, April, 1978.

"Structure and Function of a General Purpose Input/Output Processor," Alice Parker, Andrew Nagle, and James Gault, Carnegie-Mellon University, Electrical Engineering Department, unpublished paper, August, 1977-

"The Development of GLIDE: A Hardware Descriptive Language for Interfacing and I/O Port Specification," Alice Parker, Carnegie-Mellon University, Electrical Engineering Department, unpublished paper, August, 1978.

4.0 PERSONNEL SUPPORTED

Alice C. Parker, Principal Investigator

Daniel Siewiorek, Associate Investigator

Andrew Nagle, Research Assistant, MSEE, December, 1976

Louis Hafer, Research Assistant, MSEE, December, 1976

5.0 BIBLIOGRAPHY

(Barb77a) Barbacci, M.R., et. al, "Architecture Research Facility: ISP Descriptions, Simulation and Data Collection," Proceedings, 1977 National Computer Conference, Dallas, Texas, June 1977.

(Barb77b) Barbacci, M.R., Barnes, G.E., Cattell, R.G. and Siewiorek, D.P., "The ISPS Computer Description Language," technical report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1977.

(DEC72) Digital Equipment Corporation, "PDP-8/E Maintenance Manual, vol. 1, no. DEC-8E-HR1B-D, 1972.

(Hafe78) Hafer, L.J. and Parker, A.C., "Register-Transfer Level Automatic Digital Design: The Allocation Process," Design Automation Conference Proceedings, vol. 15, 1978.

(Park78) Parker, A.C. and Hafer, L.J., "The Application of a Hardware Descriptive Language for Design Automation," Proceedings of the Third Jerusalem Conference on Information Technology, August 1978.

(Snow78a) Snow, E.A., Siewiorek, D.P. and Thomas, D.E., "A Technology-Relative Computer-Aided Design System: Abstract Representations, Transformations and Design Tradeoffs," Proceedings of the 15th Design Automation Conference, Las Vegas, Nevada, June 1978.

(Snow78b) Snow, E.A., "Automation of Module Set Independent Register-Transfer Level Design," Ph.D. dissertation, Electrical Engineering Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1978.

(Thom77a) Thomas, D.E., "The Design and Analysis of an Automated Design Style Selector," Ph.D. dissertation, Electrical Engineering Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1977.

(Thom77b) Thomas, D.E. and Siewiorek, D.P., "Measuring Designer Performance to Verify Design Automated Systems," Design Automation Conference Proceedings, vol. 14, pp. 411-418, 1977.