SOME OBSERVATIONS ON SEMICONDUCTOR TECHNOLOGY AND THE ARCHITECTURE OF LARGE DIGITAL MODULES

Samuel H. Fuller and Daniel P. Siewiorek

Departments of Computer Science and
Electrical Engineering
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

August, 1973

This work was sponsored in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-73-C-0074) and in part by the National Science Foundation Grant GJ 32758X.

ABSTRACT

This report summarizes the discussion of a workshop on the Architecture and Application of Digital Modules that was held on June 7-8, 1973 at Carnegie-Mellon University. The purpose of the workshop was to identify the major influences that continuing advancements in semiconductor technology will have on the next generation of digital systems. The workshop, and this report, can be approximately partitioned into three main topics: discussion of current register-transfer level module sets and what can be learned from their development and use; the state of semiconductor technology and its current trends; and finally, discussion of current efforts to define or build computer structures that may become prototypes of the next generation of digital systems.

1. INTRODUCTION

Modules for computer system design are becoming increasingly complex, driven by decreasing cost and size of hardware and increasing computer system performance requirements. Standard modules have evolved from circuit elements to gates and flip-flops to integrated-circuit chips to register-transfer level module sets. Because of the continuing development of semiconductor technology, LSI components (e.g., memory chips with ≥ 1K bits and microprocessors) may become the standard components of digital design. Are these memory arrays and microprocessors the right set of large modules to use in the next generation of digital system design? To discuss this and related questions, a workshop on the Architecture and Application of Digital Modules was held on June 7-8, 1973, at Carnegie-Mellon University. To ensure as wide a range of perspectives as possible, participants were invited from computer manufacturers, semiconductor manufacturers, and universities. (See the appendix for the list of participants.)

The workshop, and this report, can be approximately partitioned into three main topics: discussion of current register-transfer level module sets and what can be learned from their development and use; the state of semiconductor technology and its current trends; and finally, discussion of efforts to define or build computer structures that may become prototypes of the next generation of digital systems. The final section of this report attempts to summarize the major observations of the workshop. While these observations lack a degree of quantitative precision that might be desired, they are general, qualitative statements that withstood the sometimes heated debate of the workshop.* The major purpose of this report is

*While the authors cannot accept credit for all the observations reported here, we do bear responsibility for any errors or distortions that may be present.

to make these observations available to a larger group than just the workshop participants, and to hopefully stimulate further investigation now that these statements are in black and white rather than merely circulating as folklore at informal workshops.

2. EXISTING REGISTER-TRANSFER LEVEL MODULE SETS

Several register transfer level modular systems have been developed in the last six years. By a modular system we mean a small set of modules that adhere to some intermodule communication protocol and are interconnected using a small set of rules to produce a system which performs the desired algorithm. Typically these systems are divided into a control part and a data part. The first such module set was the macromodules developed at Washington University in 1967 [Clark, et al., 1967].

Macromodules consist of a set of data and control modules that can be stacked together which defines implicit data and control interconnections between adjacent modules. Arbitrary pathways can be established by interconnecting modules with data and/or control cables. Due to the existence of several buses (or data paths) in a macromodule system a high degree of concurrency is available. The major goal of the macromodule project is to provide a set of easily used modules (as typified by the number of modules, data cables, and control sequences) that can also handle indefinite expandability (such as variable word length).

In 1971 a set of Register Transfer Modules (RTM's) became available from Digital Equipment Corporation (DEC) [Bell, et al., 1972]. RTM's were designed by DEC, whose primary goal was to look for a means of incorporating MSI in their line of module boards, and by Carnegie-Mellon University, whose primary interest was the teaching of systematic logic design. Like macromodules, RTM's use a distributed control scheme (currently there are approximately half a dozen control module types). As an economic decision, all the data modules (approximately a dozen data module types) were interconnected via a single bus. However, provision exists for RTM systems to have more than one data bus when increased performance is required.

Three other RT level modular systems were discussed at the workshop. One is a system developed at the University of Washington which is similar in concept to RTM's. However, a microprogrammed controller is used for the control part (approximately 75 chips with 100-200 nsec to execute a control step depending on the nature of the step). Data modules are developed as the need arises by specifying a module to a computer aided design package which then generates a wiring list. The major goal of this effort is to provide support for medical experiments at the University of Washington.

A set of asynchronous, distributed control modules is also being developed by MIT [Patil and Dennis, 1972]. Another effort at the University of Delaware has generalized the RTM control modules into a single universal control module (two of which can fit in a 14 pin dual in-line package) [Robinson, 1973]. Data parts are simply constructed from standard MSI chips in the University of Delaware system.

One of the major goals of all these projects is to teach systematic design of control logic. Semiconductor manufacturers currently offer a comprehensive set of data-part packages (registers, shift register, ALU's) while offering a bewildering array of SSI packages to perform control functions (RS, JK, Trigger flip-flops, etc.). By integrating these control modules into convenient and economic packages the semiconductor manufacturers could help reduce the pitfalls of conventional control logic design. Even if the control modules are not made available as chips, designers can still use the techniques typified by distributed, asynchronous control to reduce design and debugging time.

Some of the most interesting discussions at the workshop included comparisons of the cost, performance and design time of the two complete RT level modular systems versus standard SSI/MSI designs.

First, with respect to cost, macromodules and RTM's seem more expensive when compared to standard logic design. However, they owe a substantial portion of their cost specifically to those features which make them modular systems (to establish module protocol, to allow word extendability, etc.). It was estimated that this cost was 50%-70% of the total cost of macromodules and 30% of RTM's. A system built with macromodules might cost between 2 and 10 times that of a comparable system built for the same task in SSI and MSI components.

However, this extra cost is the payment necessary to achieve the design goals of flexibility, very short design time, and expan bility. The advantages of short design and debugging time in a one-of-a-kind, quick turnaround, experimental environment are obvious. It was stressed for both macromodules and RTM's that the translation of an algorithm from paper design to hardware, disregarding wiring errors, always produced a system that operated as specified.

DEC has used RTM's as a breadboarding technique to debug new approaches as well as produce low volume, custom systems where engineering design time is a major portion of the product cost. Presently, DEC has marketed over 300 custom systems that have been designed and built with RTM's. A typical system consisted of 50-100 steps, i.e., control modules; the largest system built consisted of a little over 500 steps. Most RTM systems of more than 100 steps use a ROM control unit rather than separate control modules for each step.

To date, macromodules have been used extensively in a hybrid fashion: coupled to a computer, they perform the small portion of the calculation which consumes most of the time. Comparison of performance between design with RT module sets and conventional logic is best seen by a number of examples:

1. At Carnegie-Mellon University, a PDP-8 has been built with RTM's in 55 control steps for double the cost and only 40% of the speed of a real PDP-8. (The point of this PDP-8 example is that the major area for RTM's is custom design, not general purpose computing. It is difficult to envision a modular architecture which could offset the factor of 2 in speed and cost.)

2. Matrix multiply programmed on a small machine took 400 μsec, on a CDC 7600 5 μsec and in macromodules 35 μsec.

3. The FFT (Fast Fourier Transform) butterfly multiply performed in macromodules was comparable in execution time to one programmed on the CDC 6600.

4. The major path of an electrocardiogram preprocessor took from 7 μsec (CDC 6600) to 37 μsec (PDP-9) when programmed in assembly language on a general purpose computer. A macromodule system took 3 μsec and a special purpose TTL design a projected $1\frac{1}{2}$ μsec.

The last two examples illustrate that (1) RTM's and macromodules compete successfully with general purpose computers when used in some high speed applications, since hardwired implementations of the algorithms do not incur the overhead of instruction fetch and decode, and (2) the modular systems can exploit the parallelism in the algorithm that a standard single-instruction-stream single-data-stream computer cannot.

3. SEMICONDUCTOR TECHNOLOGY

Several microprocessor chips (or small sets of chips) were described by the representatives from the semiconductor manufacturers: specifically, Intel's MCS-4 (4 bits/word) and MCS-8 (8 bits/word), National's 16 bits/word, and American Microsystems' (AMI) 16 bits/word processors. For discussion of these microprocessors see [Intel, 1972 A,B; National, 1972].

Two other microprocessors were discussed that are currently in various stages of development: Intel's 8080 and SMS's bipolar microprocessor. The Intel 8080 is an 8-bit MOS processor in a 40 pin package, 16 of which are address lines. It has 7 8-bit registers and maintains a stack in memory. Scientific Micro Systems' (SMS) is exploring the feasibility of a small (800-1000 gates) bipolar microprocessor processor with a 250 nanosecond cycle time, as compared to the MOS cycle time of about 1 microsecond. The objective is to initially design for speed and trade it for other capabilities later.

Several future trends are apparent in the semiconductor industry:

1. Since about 1960 the commercially feasible chip complexity (i.e., number of devices per chip) has roughly doubled every one to two years. In regular logic the 4K bit RAM (13,000 devices) was introduced roughly $2\frac{1}{2}$ years after the 1K bit (4000 devices) RAM. The doubling effect also holds for random logic. The 4 bit/word Intel MCS-4 microprocessor has $\sim$ 2300 devices. The Intel 8080 will be introduced about two years after the MCS-4 and will contain $\sim$ 4500 devices.

2. The regular pattern chips (e.g., memories) have about four times the density of random logic chips (e.g., processors) for the same

manufacturing complexity. For example, the Intel MCS-4 (4 bits/word) processor is about as difficult to produce as 1K-2K RAM or ~ 4G0G-8000 devices* The Intel 8080 (8 bits/word) is on the order of complexity of a 4K RAM of ~ 13*000 devices. If this relation continues to hold in coming years, we can expect to see microprocessors equivalent in complexity and cost to ~ 500 memory words (of the same size as the processor's data path), which is less than we might predict based on current minicomputer systems (i.e., 4K to 32K words).

3. The chip complexity achievable in bipolar technology usually lags MOS technology by two years. Hence MOS memories tend to be four times the size of bipolar memories. The largest MOS RAM currently available is 4K while for bipolar RAM's it is IK. Only in the area of ROM's is bipolar density comparable to MOS. Since the increase in density of bipolar technology tracks that of MOS, the present 100-200 chip bipolar minicomputers can be expected to decrease by a factor of two in chip count per year provided the semiconductor manufacturers can provide the proper chips.

4. MOS technology is approaching bipolar speeds. Currently n-channel speeds are comparable to TTL. The major constraint on speed is heat dissipation, which is limited to less than one watt/package for air cooling.

5. Production of an LSI chip, as typified by a microprocessor, is not a small undertaking. Once the architecture is specified, it is 5-10 man-years before the component is ready to go into production.

Detailed logic design, simulation, layout, initial runs, and debugging consume most of the time. Largely because of this long and costly development time, semiconductor companies look for components with a large volume market. For example, in 1972 approximately two million 1K bit MOS RAM memories were sold. Now if we contrast this with the present minicomputer market, which is on the order of 30,000 units/year, it is not difficult to understand why the semiconductor manufacturers are reluctant to develop a minicomputer on a chip. The microprocessors that have been designed are for mass markets such as personal calculators, terminals and controllers. The popularity of 4 and 8 bits/word microprocessors is largely the result of the calculator and terminal markets, respectively.

It is interesting to note that several techniques that have been used in the architecture of large computers are being employed or seriously considered for use in microprocessors. Pipelining and microprogramming are a few examples. Also, since line capacitance off-chip to on-chip may be as large as 10:1 (with subsequent decrease in speed and increase in driver capacity),on-chip memory in the form of a cache, or some other form of high-speed scratchpad, looks attractive.

4. PMS LEVEL MODULES

Given the technological trends outlined in the previous section, how do we capitalize on them in the design of future computer structures? The emergence of the microprocessors just discussed suggests that an obvious "large" control module wculd be a microprocessor. Although there has been considerable discussion of multiple processor systems in the past, there has not been the widespread application of multiple micro-, mini-, or taacro-processors systems to give us a solid foundation from which to judge microprocessors as basic modules of design* The potential for high reliability, incremental expandability,and very high throughput is clear; the problem centers around how to interconnect the microprocessors economically and program them to cooperate effectively. Although we have no easy answer to the above problems, the workshop did isolate and discuss the following efforts in multiprocessor/multicomputer design as potential prototypes of systems built from "PMS modules", i.e., LSI microprocessors and memories.

4.1. Computer Networks

One possible prototype is the computer network as exemplified by several loop systems and the ARPA network [Pierce, 1972; Farber and Larson, 1972; Roberts and Wessler, 1970]. The links between computers are fixed and messages are passed via store and forward schemes. Data is sent serially at rates of 100 to 2000 KHz; response time is on the order of 100 to 1000 milliseconds. These performance measures indicate present computer networks are too "loosely coupled" to be considered as prototypes of high performance computer structures built from PMS modules.

* Processor-Memory-Switch. For a general description of the relation of this level of description to computer structure to other levels, such as register transfer, see [Bell and Newell, 1971].

4.2. C.mmp: A Multi-Mini-Processor

C.mmp is a multiprocessor computer system currently under construction at Carnegie-Mellon University [Wulf and Bell, 1972]. It consists of up to 16 processors (modified PDP-11's) communicating through a central cross-point switch to 16 memory modules. See Figure 4.1 for an overview of the structure of C.mmp.*

Three aspects of the C.mmp project are particularly relevant to this discussion. First, C.mmp achieves a much "tighter coupling" among its processors than computer networks because it can effectively pass a data structure between processors by passing a pointer to the data via an interprocessor interrupt. Estimates indicate it will take at least 300 μsec for jobs to communicate via the interprocessor interrupt because of the need to do a context swap at the target processor.

Second, C.mmp is a standard multiprocessor system in the sense that all the processors share the same physical address space. The time to access addressable data is independent of where it resides in physical memory. However, cache memories have been proposed to exploit the "locality" of programs and hence increase the performance of the system. The cache memories would hold read-only segments for the processors. A hit in the cache would eliminate the need for a processor to send a request through the crosspoint switch to access an operand in memory. This saving could be significant since the delays through the switch are about the same as the access time of the memory (250 ns).

Finally, the address space of a standard PDP-11 (and other 16-bit mini-computers) is only 64K bytes, yet the need was immediately felt for an address space in C.mmp on the order of 2M (million) bytes. A set of relocation registers are used in C.mmp to map the smaller address space of a processor

*We use the PMS notation of Bell and Newell [1971] in this paper to describe the structure of computer systems.

into the larger physical address space of the system. The exploitation of process locality and the requirement of a larger physical address space than any of the individual processor's virtual address space are common themes we will see again in the other two systems discussed in this report.

4.3. HSM IMP: Bolt, Beranek and Newman's Multiprocessor IMP

BBN is designing a highly reliable and modular multiprocessor to replace the Interface Message Processors (IMP's) at certain ARPA network nodes. The task is special purpose and the cost is expected to be $100,000 for a 14 processor system [Heart, et al., 1973]. The structure of the HSM (High Speed Modular) IMP is shown in Figure 4.2.

One of the main differences between the HSM IMP and C.mmp is that the HSM IMP has no centralized crosspoint switch. The initial design has two memory buses (each housing part of the shared memory) and seven processor buses (each with up to four processors and a small amount of local memory). Processor buses are connected to memory buses through bus couplers that map addresses that are not references to local memory from processor buses to memory buses. As in C.mmp, a relocation - or address mapping - unit is used to translate the smaller virtual address space of the 16-bit processor (a Lockheed SUE processor in this case) into the larger physical address space of the system.

Any processor bus can be connected to any number of memory buses and any memory bus can be connected to any number of processor buses. Memory and processor buses can also be connected to an I/O bus. Hence the bus couplers constitute a distributed crosspoint switch; each processor that wants to talk to a memory is simply connected to that memory bus. The bus couplers are an interesting alternative to the centralized crosspoint switch of C.mmp. While the bus couplers provide a very modular switching scheme, they achieve this modularity through a proliferation of cables. Programs for the HSM IMP are written so that the most frequently accessed code is in local memory attached to the processor bus, and less frequently accessed code and operands are in common, shared memory along with all I/O buffers. The use of local and shared memory in the HSM IMP is in contrast with the homogeneous shared memory in C.mmp: the HSM IMP is being programmed for a specific task - message handling in the ARPA network,* while C.mmp is being developed as a general purpose computational facility.

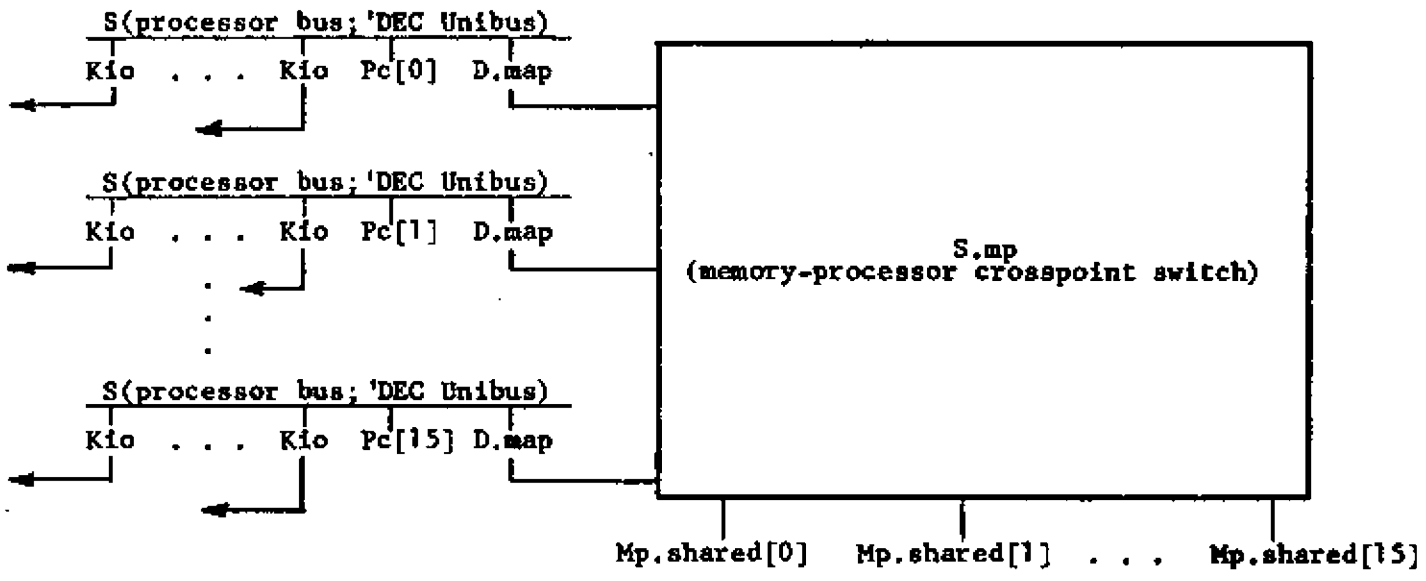
An interesting innovation in the HSM IMP is the pseudo interrupt device (PID). The PID is basic to the sequencing of tasks (or control) of the HSM IMP. Any processor can store an integer in the PID, and when the PID is "read" by any processor it returns, and then deletes, the highest integer stored. The processors use the PDI as a high speed, priority-ordered queue of pending tasks. The PID is fundamentally different from the direct processor-to-processor interrupts of C.mmp.

4.4. Computer Modules

The final scheme discussed, termed "computer modules" (CM's), is being developed at Carnegie-Mellon University [Bell et al., 1973; Fuller and Chen, 1973; Fuller, Siewiorek and Swan, 1973]. The structure of a typical CM network is shown in Figure 4.3.

Basically, CM's are processor-memory pairs with several special ports, or bus interfaces. There is no central, shared memory in the sense of C.mmp or HSM IMP (i.e., memory modules not specifically associated with any processor).

*While the HSM IMP is being developed for a specific task, it is nonetheless believed by its designers to be applicable to a wide spectrum of tasks.



S(processor bus; 'DEC Unibus)
Kio . . . Kio Pc[0] D.map
S(processor bus; 'DEC Unibus)
Kio . . . Kio Pc[1] D.map
S(processor bus; 'DEC Unibus)
Kio . . . Kio Pc[15] D.map
S.mp
(memory-processor crosspoint switch)
Mp.shared[0]
Mp.shared[1]
. . .
Mp.shared[15]

Figure 4.1 The General Structure of C.mmp

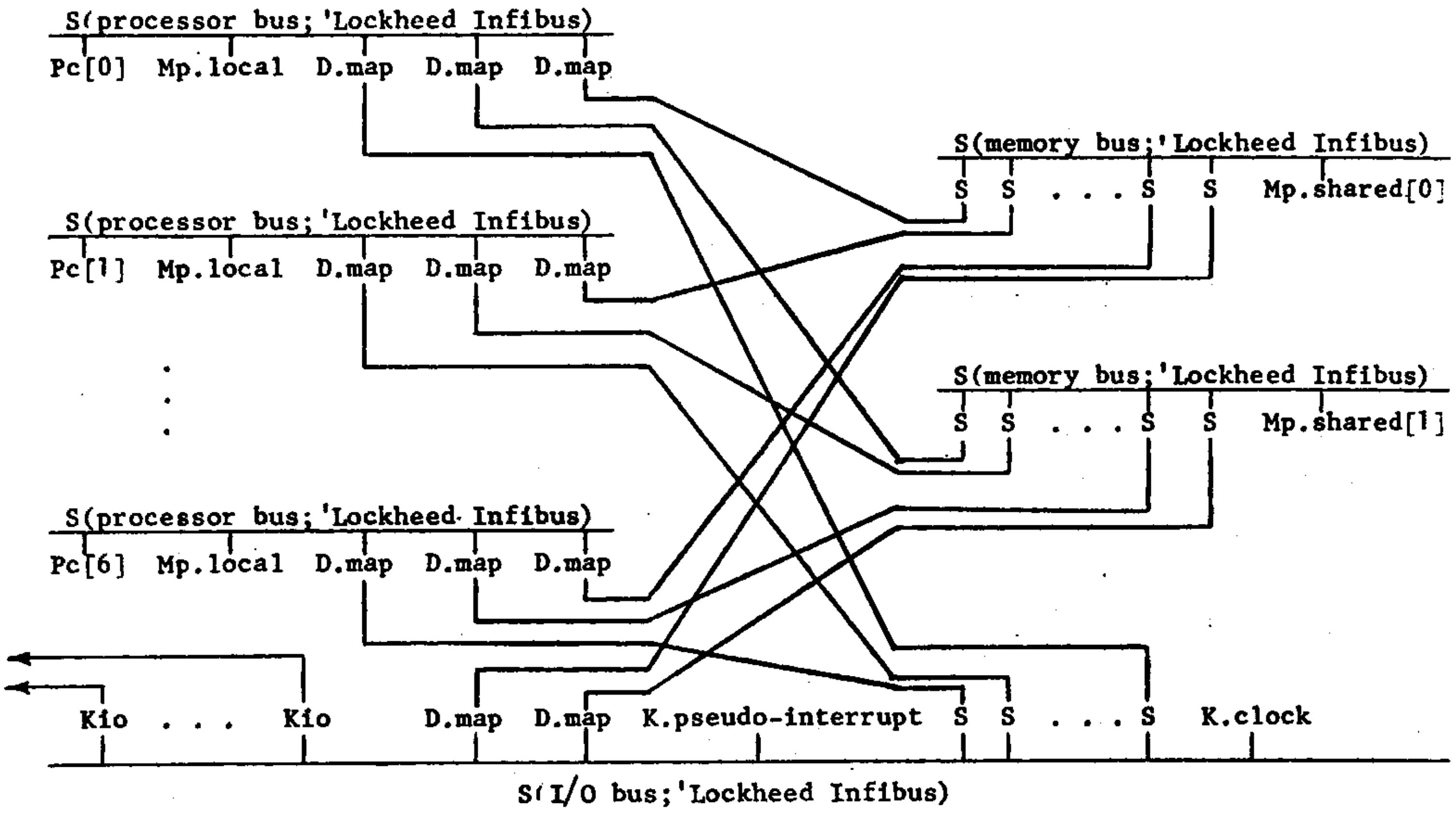

S(processor bus; 'Lockheed Infibus)
Pc[0] Mp.local D.map D.map D.map
S(processor bus; 'Lockheed Infibus)
Pc[1] Mp.local D.map D.map D.map
S(processor bus; 'Lockheed Infibus)
Pc[6] Mp.local D.map D.map D.map
S(memory bus; 'Lockheed Infibus)
S S . . . S S Mp.shared[0]
S(memory bus; 'Lockheed Infibus)
S S . . . S S Mp.shared[1]
Kio . . . Kio D.map D.map K.pseudo-interrupt S S . . . S K.clock
S(I/O bus; 'Lockheed Infibus)

Figure 4.2 The General Structure of BBN's HSM IMP

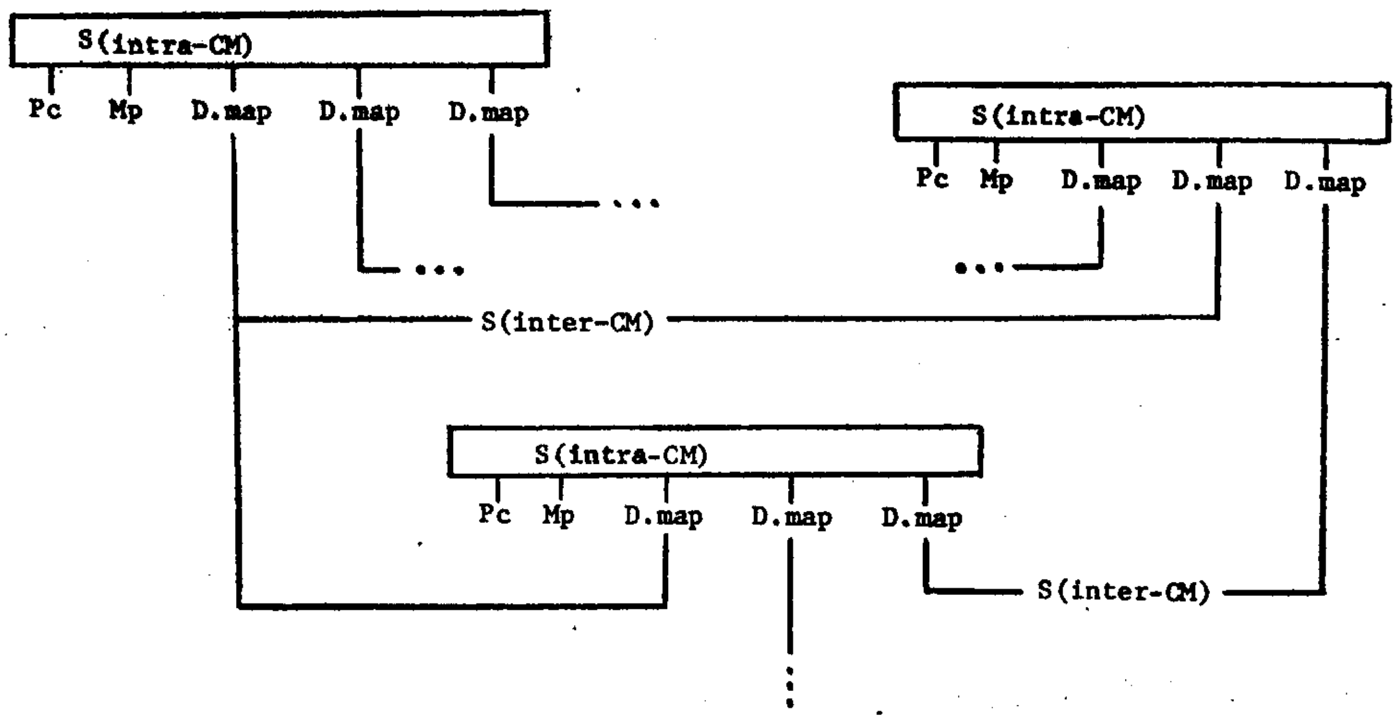


Figure 4.3 The General Structure of a Computer Module System

The physical address space in a CM system is the sum of the local memories of the CM's making up the system (Figure 4.3). As in C.mmp and the HSM IMP, each CM processor has a small, virtual address space (64K bytes) and a mapping unit (in this case the bus interface) that translates virtual addresses into the large physical address space. The bus interfaces, or simply D.map's, provide access to inter-CM buses. A D.map monitors the intra-CM bus for addresses that are within segments tagged for translation. Upon recognizing such an address, the D.map maps it into the inter-CM bus address space. Similarly, D.maps may also monitor the inter-CM bus and upon recognizing an address, map it into the intra-CM bus address space. Thus a CM can request an address, and if the mapping registers are set appropriately, map across several inter-CM buses (and through several CM's) before reaching the desired word of physical memory. Whereas computer networks need cooperation from remote processors to send a message, a processor in a CM can access a remote CM's memory without the cooperation of the remote processor.

5. SUMMARY OF MAJOR OBSERVATIONS

The following observations are an attempt to state the major conclusions of the discussion at the workshop. These are not meant to be a comprehensive set of comments on RT-level modules, semiconductor technology, or PMS-level modules, but only those observations that were vague or controversial enough to warrant discussion at the workshop.

5.1. RT-Level Modules

1. Semiconductor manufacturers currently provide an adequate, and growing, set of RT-level (i.e., MSI) components to handle the standard data operations such as storage, addition, shifting, etc. However, there

is a perplexing lack of RT-level control components to handle control operations» This cannot be excused for lack of understanding of RT-level control components. Bell et al. [1972], Clark et al. [1967], Dennis and Patil [1972], and Robinson [1973], all have demonstrated workable sets of control modules.

2. The "overhead" in hardware required to transform a unit of logic into a module that observes a practical inter-module protocol is commonly on the same order of cost and complexity as the original logic. In many cases this is a small price to pay for the drastic reduction in design time. In any event, this factor should be kept in mind as future sets of modules, and future applications of modular systems, are considered.

5.2. Semiconductor Technology

1• The complexity of practical semiconductor components is doubling every one to two years. The industry's current limits in MOS manufacturing ability are chips that contain 4K bit random access memories or 8 bits/word microprocessors.

2. Random logic components (e.g., microprocessors) have consistently followed regular logic components (e.g., memories) by a factor of four in complexity. One consequence of this is that a 4 or 8 bit microprocessor is roughly equivalent to 500 4 or 8 bit words of random access memory, respectively.

3. A semiconductor chip that has the potential sales volume of the current minicomputer market, i.e., about 30,000 units/year, would

not be economically feasible to produce. The major consequence of this is that microprocessors in the foreseeable future will be designed for such mass markets as personal calculators and intelligent terminals.

5.3. PMS-Level Modules

An observation from current developments in the semiconductor industry is that small microprocessors are the most obvious LSI control module. The following comments concern the problems of building computer structures with microprocessors, and other LSI components, e.g., random access memories and read only memories.

1. There have been significant efforts in the past to decompose algorithms into parallel processes. We know how to parallelize at a small grain (arithmetic expressions in the 360/91 at the instruction level) and a large grain (tasks in a multiprogramming system at the several 100's to 1000's instruction level). At the intermediate level of problem granularity there has been little progress made with a general solution. However, a number of specific and important applications have been studied and are known to decompose efficiently into parallel tasks, e.g., weather simulation, signal processing, airline reservation systems, message switching, and many vector and string processes. Since a number of the applications that can be decomposed into parallel processes are sufficiently important, they justify work in multiple processor systems and encourage work in the development of parallel algorithms for other applications.

2. Multiple microprocessor systems should have some form of local memory and attempt to exploit any locality present in jobs to minimize the inherent switching delays associated with multiple processors accessing a central, shared memory. In special purpose tasks, such as an IMP, an a priori analysis of the code can identify the commonly used segments of a program; in a general purpose application some automatic, dynamic scheme (such as the C.mmp cache proposal) must be used.

3. Computer structures will often require a physical address space much larger than the virtual address space of an individual microprocessor. Some convenient, high performance method must be used to provide a mapping from the small microprocessor address space to the larger physical address space.

4. Inter-(micro)processor communication is one of the least understood issues in multiprocessor systems. Hopefully experience with the various intercommunication schemes in C.mmp, HSM IMP, CM's, and other multiprocessor structures will provide a basis for further work in this area.

ACKNOWLEDGMENT

We gladly acknowledge the assistance of Raj Reddy. He initially suggested the idea of this workshop and provided much of the energy and work required to make it a reality.

References

Bell, C. G., R. C. Chen, S. H. Fuller, J. Grason, S. Rege and D. Siewiorek. "The Architecture and Applications of Computer Modules: A Set of Components for Digital Design," IEEE Computer Society International Conference, COMPCON 73, March 1973, 177-180.

Bell, C. G., J. L. Eggert, J. Grason and P. Williams. "The Description and Use of Register-Transfer Modules (RTM's)," IEEE Trans. on Computers, Vol. C-21, No. 5, 495-500, May 1972.

Bell, C. G. and A. Newell. Computer Structures: Readings and Examples, McGraw-Hill, New York, New York, 1971.

Clark, W. A., S. M. Ornstein, M. J. Stucki, A. S. Blum, T. J. Chaney, R. E. Olsen, R. A. Dammkoehler, W. E. Ball, C. E. Molnar and A. Anne. "Macromodular Computer Systems," AFIPS Conference Proc., Vol. 30, SJCC 1967, 335-402.

Dickson, C. "A Macromodule User's Manual," Computer Systems Lab. Tech. Report 25, Washington University, St. Louis, Missouri, 1973.

Ellis, R. and M. A. Franklin. "High-Level Logic Modules: A Qualitative Comparison," IEEE Computer Society International Conference, COMPCON 72, September 1972, 313-316.

Farber, D. J. and K. C. Larson. "The System Architecture of the Distributed Computer System -- The Communications System," Symposium on Computer Networks, The Polytechnic Institute of Brooklyn, April 1972.

Fuller, S. H. and R. C. Chen. "The I/O Port Architecture for Computer Modules," Departments of Computer Science and Electrical Engineering Technical Report, Carnegie-Mellon University, Pittsburgh, Pa., March 1973.

Fuller, S. H., D. P. Siewiorek, and R. J. Swan. "Computer Modules: An Architecture for Large Digital Modules," Depts. of Computer Science and Electrical Engineering Technical Report, Carnegie-Mellon University, Pittsburgh, Pa., Sept. 1973.

Heart, F. E., S. M. Ornstein, W. R. Crowther, and W. B. Barker. "A New Minicomputer/Multiprocessor for the ARPA Network," Proc. AFIPS NCC 42, 1973, 529-537.

Intel Corporation, MCS-4 Micro Computer Set Users Manual, Santa Clara, Calif., July 1972.

Intel Corporation. MCS-8 Micro-Computer Set Users Manual, Santa Clara, Calif., November 1972.

National Semiconductor. "Introduction to the Microprogram for Implementation of IMP-16 Macroinstruction Set," Pub. No. 42000 10A, Santa Clara, Calif., June 1972.

National Semiconductor. "Product Description: General Purpose Controller/Processor (GPC/P) MOS/LSI System Kit," Pub. No. 42000 05A, Santa Clara, Calif., March 1972.

Newell, J. A. "Data and Control Signal Distribution in a Macromodular Data-processing Manifold," Computer Systems Lab. Tech. Report 21, Washington University, St. Louis, Missouri, 1973.

Patil, S. S. and J. B. Dennis. "The Description and Realization of Digital Systems," IEEE Computer Society International Conference, COMPCON 72, September 1972, 223-226.

Pierce, J. R. "How Far Can Data Loops Go?," IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972, 527-530.

Roberts, L. G. and B. D. Wessler. "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proc., Vol. 36, SJCC 1970, 543-549.

Robinson, D. M. "Digital System Design with Control Modules," IEEE Computer Society International Conference, COMPCON 73, February 1973, 207-210.

Wulf, W. A. and C. G. Bell. "C.mmp -- A Multi-Mini-Processor," AFIPS Conference Proc., Vol. 41, Part II, FJCC 1972, 765-777.

Wulf, W. A., E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, F. Pollack. "HYDRA: The Kernel of a Multiprocessor Operating System," Department of Computer Science Technical Report, Carnegie-Mellon University, Pittsburgh, Pa., June 1973.

APPENDIX

LIST OF WORKSHOP PARTICIPANTS

D. Baker, Digital Equipment Corporation, Maynard, Mass.
C. Barry, Washington University, St. Louis, Missouri
G. Bell, Digital Equipment Corporation, Maynard, Mass.
D. Casasent, Carnegie-Mellon University, Pittsburgh, Pa.
H. Chuang, Washington University, St. Louis, Missouri
D. Clark, Xerox Research Center, Palo Alto, Ca.
W. Clark, Consultant, Boston, Mass.
J. Cox, Washington University, St. Louis, Missouri
S. Culp, Texas Instruments, Houston, Texas
E. Davidson, Stanford University, Stanford, Ca.
J. Eggert, Digital Equipment Corporation, Maynard, Mass.
R. Ellis, Washington University, St. Louis, Missouri
S. Fuller, Carnegie-Mellon University, Pittsburgh, Pa.
J. Grason, Carnegie-Mellon University, Pittsburgh, Pa.
T. Hoff, Intel Corporation, Santa Clara, Ca.
A. Jordan, Carnegie-Mellon University, Pittsburgh, Pa.
T. Kehl, University of Washington, Seattle, Wash.
B. Lampson, Xerox Research Center, Palo Alto, Ca.
J. Lehmann, National Science Foundation, Washington, D.C.
V. Lesser, Carnegie-Mellon University, Pittsburgh, Pa.
H. McFarland, American Micro-Systems, Inc., Santa Clara, Ca.
C. Molnar, Washington University, St. Louis, Missouri
A. Newell, Carnegie-Mellon University, Pittsburgh, Pa.
G. Oliver, Scientific Micro Systems, Mountain View, Ca.
S. Ornstein, Bolt, Beranek and Newman, Cambridge, Mass.
S. Patil, MIT, Cambridge, Mass.
J. Raymond, Texas Instruments, Inc., Houston, Texas
R. Reddy, Carnegie-Mellon University, Pittsburgh, Pa.
T. Reichert, Carnegie-Mellon University, Pittsburgh, Pa.
G. Reyling, National Semiconductor Corporation, Santa Clara, Ca.
D. Robinson, University of Delaware, Newark, Delaware
F. Rosenberger, Washington University, St. Louis, Missouri
D. Siewiorek, Carnegie-Mellon University, Pittsburgh, Pa.
W. Smith, Digital Equipment Corporation, Maynard, Mass.
M. Stucki, Washington University, St. Louis, Missouri
S. Teicher, Digital Equipment Corporation, Maynard, Mass.
C. Thacker, Xerox Research Center, Palo Alto, Ca.
W. Wulf, Carnegie-Mellon University, Pittsburgh, Pa.