SOME OBSERVATIONS ON MODULAR DESIGN TECHNOLOGY AND THE USE OF MICROPROGRAMMING

D.P. Siewiorek
M.R. Barbacci
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

July 1974

ABSTRACT

As modules become more complex the advantages and disadvantages of modularity have become more pronounced. The cost of modularity is measured not only in added hardware but also in a loss of flexibility. Functions that are easy to implement at a submodule level may be very difficult, or even impossible, to duplicate at the modular level. We term this a loss of transparency. The added hardware and transparency costs are given for existing module sets and projections made for the next generation of modules. Finally, a microprogrammed implementation of the control portion of existing and projected module sets is shown to be a way to decrease hardware costs and increase transparency.

This paper will be published in the Infotech State of the Art Report on Microprogramming and Systems Architecture.
This work was supported in part by the Advanced Research Projects Agency (ARPA) of the Department of Defense, under contract F44620-73-C-0074, monitored by the Air Force Office of Scientific Research and in part by the National Science Foundation under grant GJ 32758X.

1 - INTRODUCTION

Increased attention has been focused on the understanding of the design process as systems, whether physical or social, become more complex. The sheer complexity of many systems demands an orderly design approach. Freeman and Newell [Freeman, 1971] use the term "functional design" to describe the tendency of humans to design in terms of functions. A paraphrase of their basic model of a design task environment consists of a set of modules and a set of functions such that:

1) Each module provides a set of functions

2) For each function it provides, a module requires a set of functions

3) A functional connection can occur between two modules if one provides a function required by the other

4) A super-module consists of a set of modules (its parts) and a set of functional connections between them such that: (a) The functions provided are those provided by the modules that are not consumed in functional connections, and (b) The functions required are those required by the modules that are not provided by a functional connection.

In other words, "functional design" is a hierarchical process. Modules at one level are used to construct super-modules at a higher level, which may in turn be used to construct super-super-modules, and so on. To take an example from current computer technology: chips are interconnected to make boards, boards are plugged into back panels to make system units, and system units are plugged into cabinets to form a system.

The advantages of having a standard set of modules at any given level are well

documented [Bell, 1973; Davidow, 1972; Parnas 1971] and a partial list might include:

1) Reduced development time by allowing the design task to be partitioned, making better use of the resources (time and manpower)

2) Increased flexibility by allowing the alteration of specifications and the redesign of components

3) Comprehensibility by allowing the students of the system to concentrate in well defined pieces of the final object

4) Maintainability by allowing the identification of faulty components which can then be replaced or repaired

5) Economy of scale by mass producing a small number of standard modules rather than supporting custom design of modules

6) Significantly decrease system construction or modification time by conducting all design with high level modules as primitives. This is different from (1) and (2). Here we are talking about building systems with predefined modules, while (1) and (2) are concerned with the specification of the original modules.

On the other hand modularity also has disadvantages, not often stated:

1) Rigid intramodule connections may result in both suboptimal use of resources and suboptimal performance

2) There is an overhead incurred by simply making a collection of functions part of a module set

3) Functions that are easy to derive given some basic modules may be extremely difficult or impossible to achieve at the supermodule level. We term this a ***loss of transparency***.

This paper will survey the evolution of technology and its impact on modular design. Existing module sets will be reviewed and some quantitative result will be given on the advantages and disadvantages of modular design outlined above.

Microprogramming in modular systems will be shown to be a cost effective substitute for random control logic and a means to minimize the loss of transparency. Finally some predictions will be given on the future shape of modules and the cost of modular design.

2 - DIGITAL FUNCTIONS AND THE DIGITAL DESIGN HIERARCHY

In this section the "functional design" process described by Freeman and Newell [Freeman, 1971] is given in terms of digital systems design. First the functions are described and subsequently the hierarchy of design levels.

Bell and Newell [Bell, 1971] have identified seven basic component types in terms of functions:

> Memory (M) components hold or store information over periods of time
>
> Links (L) components transfer information between components in a system
>
> Control (K) components evoke the operation of other components in a system
>
> Switch (S) components construct links between other components
>
> Transducer (T) components change the encoding of information
>
> Data-operation (D) components produce information with new meanings
>
> Processor (P) components are capable of interpreting a program to execute a sequence of operations.

Modules with functions of these types can be interconnected into super-modules that provide one or more of these functions. This hierarchy of digital design levels is given by Bell and Newell [Bell, 1971] as:

> Circuit level: Circuits are build by interconnections of the basic components (diodes, transistors, resistors, etc.) according to electrical circuit laws. The behavior of the components is described in terms of voltages and currents, continuously varying through time.
>
> Switching circuit level (Sequential and Combinational sublevels): This level is

unique to digital systems design (the circuit level is shared with the rest of electrical engineering). The system structure is given by a collection of gates and flip-flops, and the behavior by a set of Boolean equations. This has been the traditional level of digital design.

Register Transfer (RT) level: A combination of switching circuits is used to form registers, functional units, and data paths, to perform register transfers and other operations.

Programming level: Although not part of the traditional domain of hardware design, it is included in the hierachy because of its unique association with digital computers. The basic components are the interpretation cycle, the machine instructions, and the data operations (which are defined at the RT level).

PMS level: This level owes its name to the main components, Processors, Memories, and Switches. This level depicts a digital system as a continuously operating network of components, sharing this view with the switching circuit level.

3 - THE EVOLUTION OF TECHNOLOGY AND MODULE SETS

As technology has evolved the fundamental modules have become more and more complex. Standardized module sets have evolved from circuit elements, to gates and flip-flops (Small Scale Integration) to register transfer level modules sets (Medium Scale Integration). At each stage in the evolution, technology has dictated what components are sufficiently cheap to fabricate that they (i.e., their functions) can be considered expendable. For example, in the late 1950's considerable attention was devoted to minimizing the number of gate inputs to save the cost of diodes (gates were built out of circuit elements). With the advent of Small Scale Integration (SSI), gates are available only with a specified number of inputs. If a three input gate is desired it is often acceptable to use a four input gate. New constraints such as chip types (package), unused portions of a chip, or chip position on a board were more important than minimizing gate inputs. Table 1 gives examples of expendable components at different levels. Justification of the expendable components for the RT level and above will be given in subsequent sections.

level of primitive module	example of expendable component
Switching Circuit	gate input
Register Transfer	ALU function
PMS	Sequences of register transfer operations, Processors

Table 1. The expendable components

Through SSI technology, modularity had many advantages and few disadvantages since SSI modules provided essentially the most primitive functions required in digital systems design. Digital signals were the norm and there was little need to go below the gate level to circuit components and their analog signals. Some inflexibility was introduced by SSI level modules. For instance, the AND function of 14 variables would be implemented as a tree-like network of smaller AND gates. These inefficiencies were, in general, tolerable.

With Medium and Large Scale Integration module complexity has increased dramatically and the advantages and disavantages of modularity have become more pronounced. The next section surveys existing module sets at the register transfer level.

4 - EXISTING REGISTER TRANSFER LEVEL MODULE SETS

RT level components are structures of logic level components and the operations tend to be more general than at the logic level. These operations range from simple transfers, to logical expressions, to arithmetic expressions. Typically RT level module sets are divided into a control part and a data part. The appearance of control introduces a discreteness in space by allowing the specification of selected components that are performing the register transfer operations of interest at any point in time. This is clearly an abstraction since the modules that are used to build the RT components are active all the time. This abstraction allows us to concentrate on those components that are changing values.

The first such module set was the macromodules developed by Washington University in 1967 [Clark, 1967]. Macromodules consist of a set of data and control modules that are stacked together and interconnected via bus 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 was to provide a set of easily used modules that could handle indefinite expandability (as typified by variable word length).

In 1971 a set of Register Transfer Modules (RTM's) became available from Digital Equipment Corporation (DEC) [Bell, 1972a, 1972b]. RTM's were designed by DEC, whose primary goal was to look for a means of incorporating Medium Scale Integration (MSI) in their line of module boards, and 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. Figure 1 depicts the RTM implementation of a system to sum the integers from 1 to N.

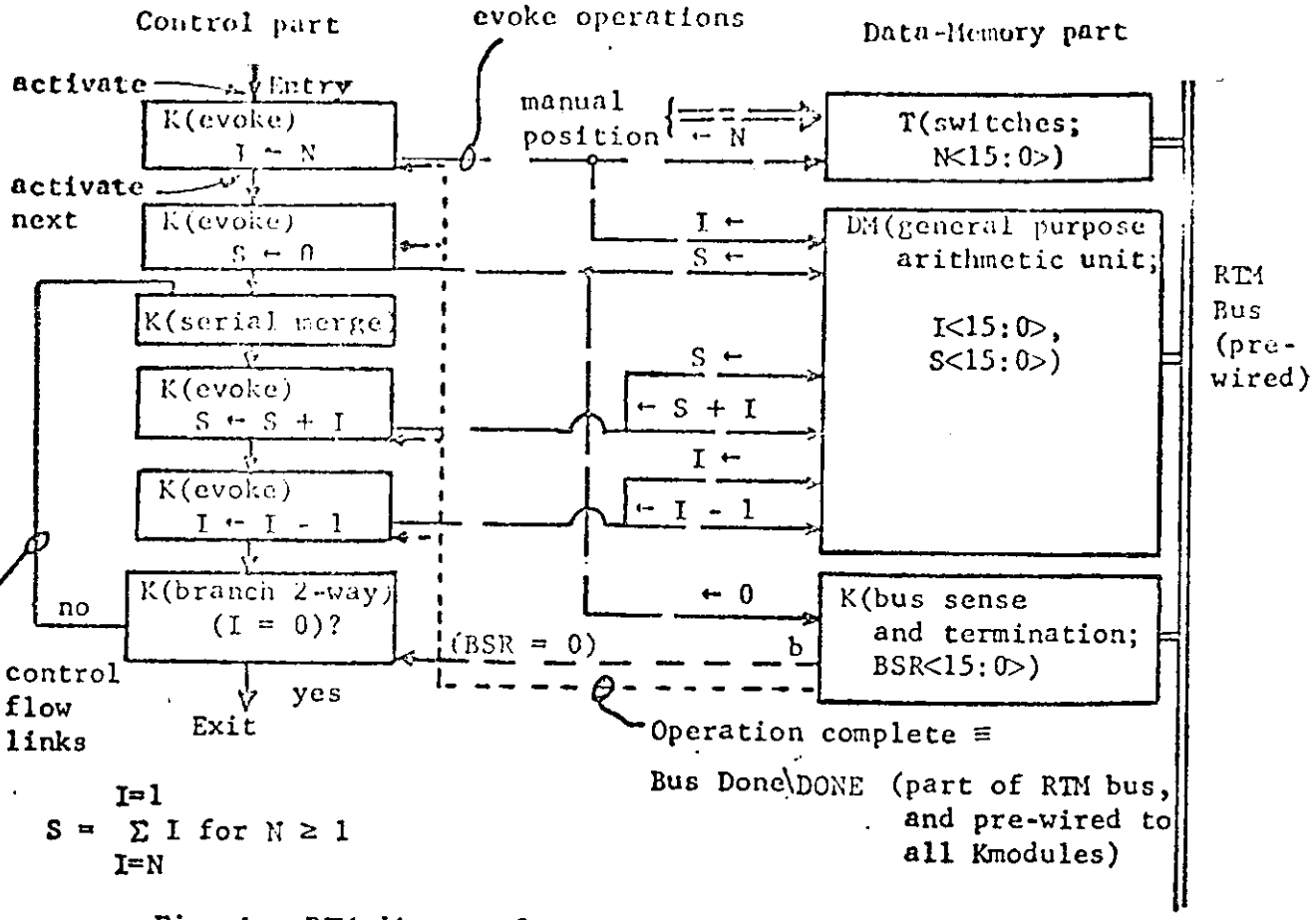


Fig. 1. RTM diagram for sum of integers from 1 to N.

The connections shown in the figure are all that are required to construct the system. Note that the primitive functions are very similar to those available at the assembly

language level of programming. Other RT level module sets are being developed at MIT [Patil, 1972], the University of Washington, and the University of Delaware [Robinson, 1973]. An interesting feature of the latter module set is that the data part is composed solely of commercially available MSI chips.

The advantages of design with these module sets are dramatic. A PDP-8 like minicomputer could be designed and built in 6-7 man-months using discrete components. A similar processor built from SSI components might take 2-3 man-months and from MSI/LSI components about one man-month to design and construct [Bell, 1974]. A PDP-8 like minicomputer has been designed, constructed, and debugged with RTMs in 8-10 man-hours. As in the case with all the RT module sets, the translation from paper design to hardware implementation is a one-for-one process. A large majority of the systems work the first time power is applied. The module sets provide a very clean intermodule communications protocol which eliminates any timing problems. At least one company (DEC) has used a RT level module set as a breadboarding technique to debug new products as well as for production of low volume, custom design items 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 build with RTM's.

When employing more complex modules as primitives it becomes feasible to explore many alternative design for a given specification. Historically design automation programs have primarily served in a bookkeepping capacity in the implementation of a design. The PDP-8 built from RTM's required about 55 control modules and 10 data modules as contrasted with several hundred chips and an ad hoc controller for a real

PDP-8. The smaller number of components plus the regularity of the design process with RTM's has enabled the construction of a program called EXPL [Barbacci, 1973] which takes the description of an algorithm in a RT level language, ISP [Bell, 1971], and some cost-time constraint as inputs. EXPL produces a near optimal RTM solution as output by use of graph tranformations and heuristic search techniques. Another effort [Rege, 1974] explores the space of data part designs, producing optimal allocations of operators (physical components) to the data operations of a sequential flowchart.

To obtain these advantages a price is paid. A design using module sets tends to be slower and costlier in terms of hardware than a comparable system designed with SSI and MSI components. The RTM PDP-8 cost twice as much and ran at 40% of the speed of the real PDP-8. A system designed with macromodules might cost between two and ten times more than a comparable MSI/SSI system. The extra cost is due to the overhead of making a module part of a module set (e.g., to establish control protocols, to allow word extensibility, to permit physical connections, etc.) This overhead is approximately 30% for RTM's and 70% for macromodules [Fuller, 1973a].

Although module sets may be slower than comparable systems built from lower level primitives, they are extremely competitive with general purpose computers. An algorithm can be hardwired with modular components and not incur the overhead of fetching and decoding instructions. Also, the modular implementation can take advantage of any parallelism in the algorithm. For example the macromodule implementation of certain algorithms were ten times faster than programs on a PDP-9 and between 1 and 2 times faster than a CDC 6600 [Fuller, 1973a].

The component interconnection rules that define a RT level module set can also contribute to the increased cost of a modular implementation. These rules can potentially lead to gross inefficiencies, inefficiencies that should be carefully weighted in future module sets. To formalize this notion we will introduce the concept of ***transparency***. A module suppresses some of the detail of its constituent components and their interconnections while providing a set of functions to the user. This suppression of detail is both the strenght and the weakness of a modular approach. It is a strenght in that a designer can conceptualize and construct at a higher level. Advantages stem from having a smaller set of components and facts to keep in mind. A weakness arises if some required function, which the constituent modules can perform, is not available to the user. This we term as a ***loss of transparency*** [Parnas, 1972]. The missing function may be very difficult or even impossible to reproduce at the module level. For example, RTMs use four four-bit ALU (Signetics SN74181) in their arithmetic element (DMgpa). The ALU is capable of performing 16 arithmetic and 16 Boolean functions of two parameters yet only 12 of these are usable at the module level. As another example, consider BCD arithmetic where the carry from each BCD digit is required. Only the carry from the most significant four-bit ALU is available to the user in RTMs. Thus to perform arithmetic on four digit, packed BCD numbers in RTMs a penalty of a factor of 4-5 in speed is paid over a module set in which the carry bits from each four-bit ALU is available.

As noted before, a primary advantage of RT level module sets is systematic design. Semiconductor manufacturers currently offer a comprehensive set of data part modules (e.g., Texas Instruments catalog, Signetics catalog) and these even form the data

part of one RT level module set [Robinson, 1973]. However, there is only a bewildering array of SSI components to perform control functions. The systematic control elements of the RT module sets are one solution to systematic design although the distributed, unary encoding of control states can be quite costly in terms of hardware for large systems. Accordingly, the RTM module set has a programmed control sequencer (Kpcs) module to evoke the various register transfer operations [Bell, 1972b]. The Kpcs is depicted in Figure 2. The control memory contains an encoded version of the control part of a RTM system. Figure 3 shows the microinstruction format. The Kevoke microinstruction selects one of six decoders and one of 32 outputs from the decoder subsequently initiating the corresponding operation. The branch microinstruction selects one of 30 input conditions for determining whether a branch should be taken. The subroutine call pushes the current value of the program counter onto the subroutine stack and jumps to the specified address. The subroutine return pops the stack into the program counter. The interpreter is very similar to a conventional processor. It fetches, decodes, and executes the microinstructions sequentially.

Figure 4 shows the relative cost of the distributed control systems and the Kpcs as a function of the number of steps. The discrete steps in the cost function of the Kpcs are caused by having to add control memory in discrete blocks (256 words). Another interesting comparison is the number of control memory bits needed to replace a gate in the distributed system. Assuming that out of every 20 control steps there is one subroutine call/return, five branches/merges, and 14 evoke operations we need 27 control memory words or 216 bits.

In the distributed control system a subroutine call/return module requires about

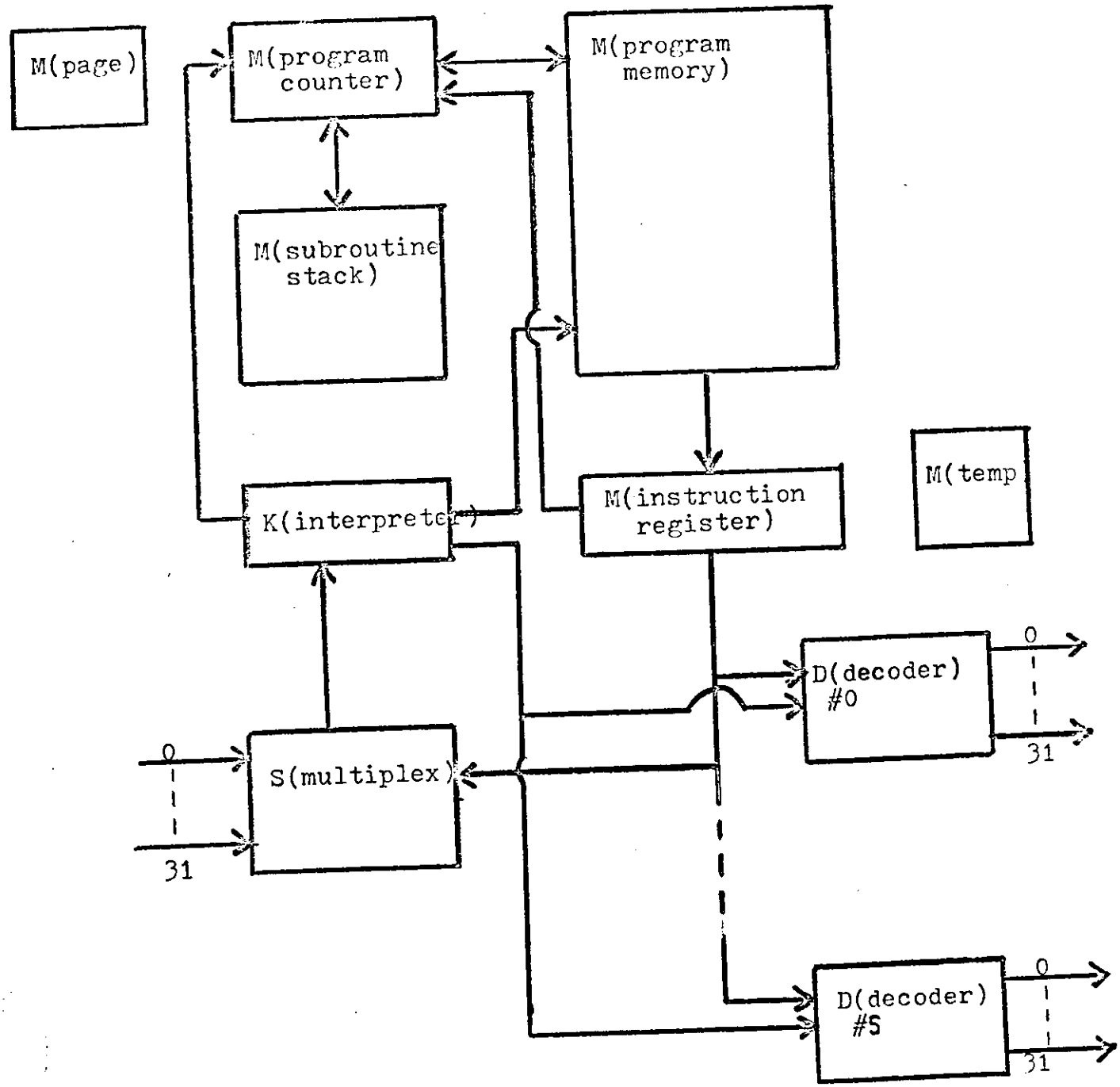
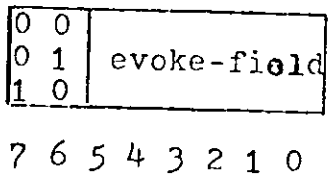
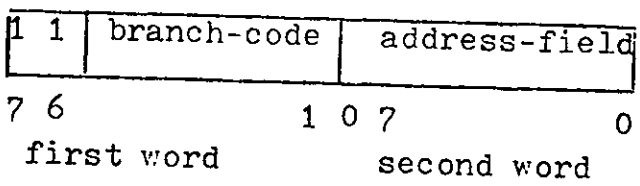


Figure 2. Kpcs diagram

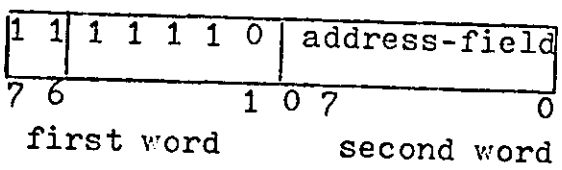
Kevoke



Kbranch



Ksubroutine call



Ksubroutine return

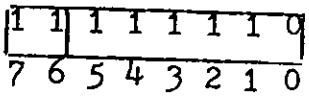


Figure 3. Microinstruction formats for Kpcs

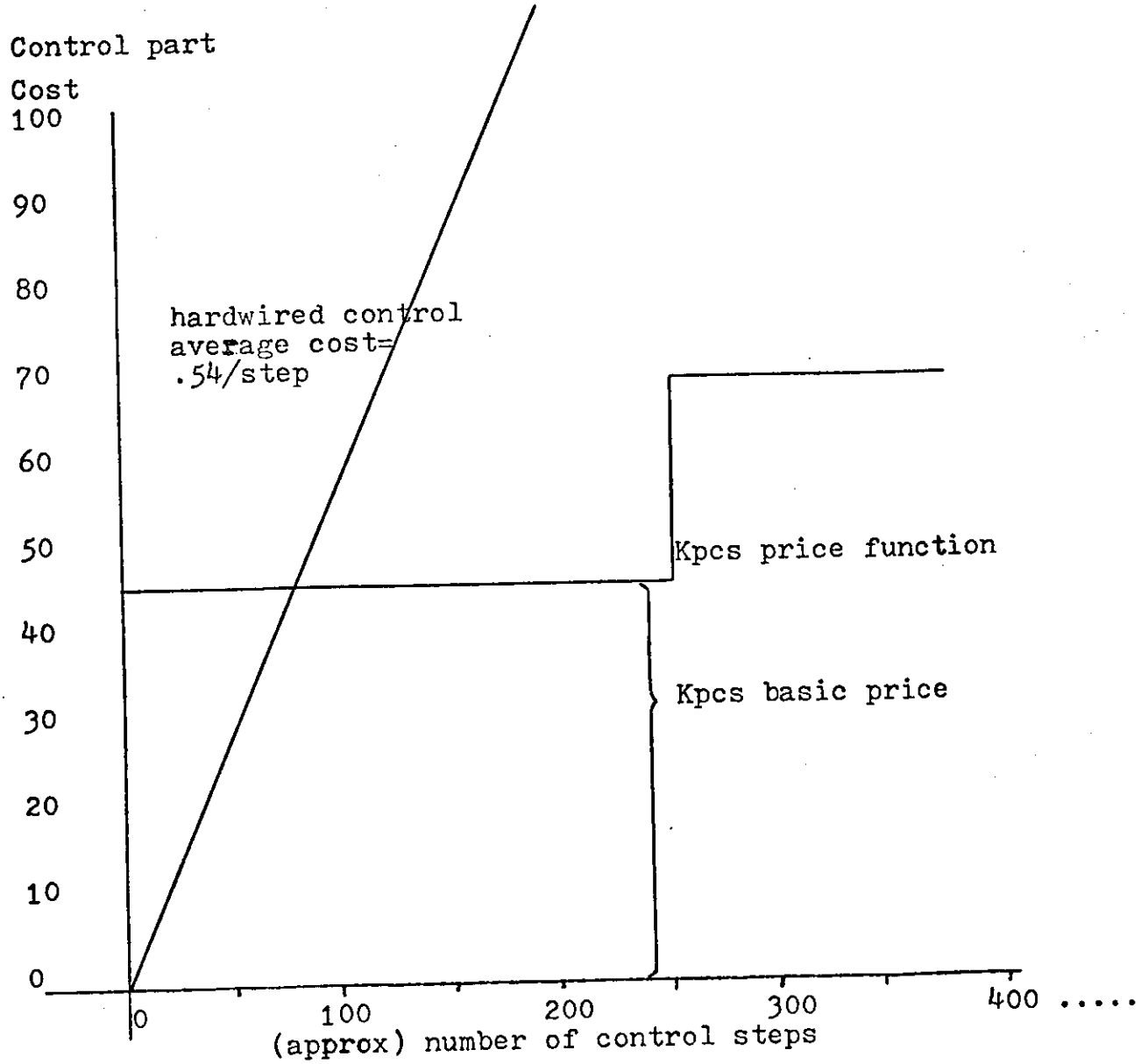


Figure 4. Cost of control part versus control steps for hardwired and Kpcs cases

10 gates, a branch/merge nine gates, and an evoke eight gates for a total of 167 gates. Thus about 1.3 bits of control memory are needed to replace a gate.

It should be noted, however, that a unary encoded, distributed control does not

make efficient use of logic gates (i.e., they can be replaced by control memory bits on an almost 1-1 ratio). A more realistic guess would be 5 to 12 control bits to replace a gate [Davidow, 1972]

In summary, microprogramming appears to be a cost effective replacement for random logic for control of large modular systems. The next section will make a few projections about the shape of future modules.

5 - FUTURE MODULE SETS

The advent of Large Scale Integration (LSI) technology has made the chip a natural boundary for a module. Currently a 4K bit MOS random access memory (RAM) or an eight bit MOS microprocessor are available on a single chip. Since 1960 the optimum chip complexity has doubled every one to two years [Fuller, 1973a]. There is no indication that this trend should not continue for the next several years. Thus in 4-6 years a 16-bit microprocessor with 1K words of memory could be available on a chip. What should a module look like when it is of this complexity?. When placed on a chip memories, due to their regular patterns, can be four times as dense as random logic [Fuller, 1973a]. This strongly suggests that a microprogrammable controller be employed. The ability to alter control sequences (i.e., programming) also implies that the majority of the functions internal to the module will be available to the user thus reducing the loss of transparency.

Assuming that the modules of the future are microprogrammable processors with associated memory some interesting conclusions can be drawn. First, by observing the Intel MCS-4 and MCS-8 [INTEL, 1972a, 1972b] it appears that a current microprocessor is equivalent in complexity and cost to ~500 memory words (of the same width as the processor's data path). Extrapolating this to microprogrammable processors it can be seen that for modules with even moderately size memories (4-8K) the processor is expendable since the dominating cost is that of the memories. Further, a single, versatile module would be more attractive to the semiconductor industry than a module

set consisting of 10 or more modules since a successful chip needs a sales volume of ~2 million units/year [Fuller, 1973a]. This is in contrast with the current minicomputer market of 30,000 units/year.

In order to achieve high computational power with these modules a mechanism must be provided for efficient communications between modules. The I/O structure of a conventional processor would be a poor way to conduct intermodule communication since a response to an interrupt might take tens of microseconds. Intermodule communication schemes based on shared memory have been studied in connection with multiprocessors [Wulf, 1972; Heart, 1973] and might be a viable alternative.

One prototype intermodule communication scheme for PMS level modules based on shared memory is the one employed by Computer Modules [Bell, 1973; Fuller, 1973b]. The structure of a typical CM network is shown in Figure 5. A CM consists of a small processor as the primary control element, local memory, and several I/O ports for interconnection to other CMs. An essential feature of CMs is the interconnectability of CMs into networks via inter-CM buses, each with a unique address space to enable efficient memory sharing. An address generated by a processor can be mapped onto its local memory, or mapped onto the address space of one of the inter-CM buses were it is recognized by the I/O port of another CM and directed to the latters local memory. A CM can also be used as a switch. An inter-CM bus address recognized by a CM I/O port can be remapped to another inter-CM bus rather than directed to local memory. Hence a CM can share memory with any other CM in the network without necessitating a direct connection.

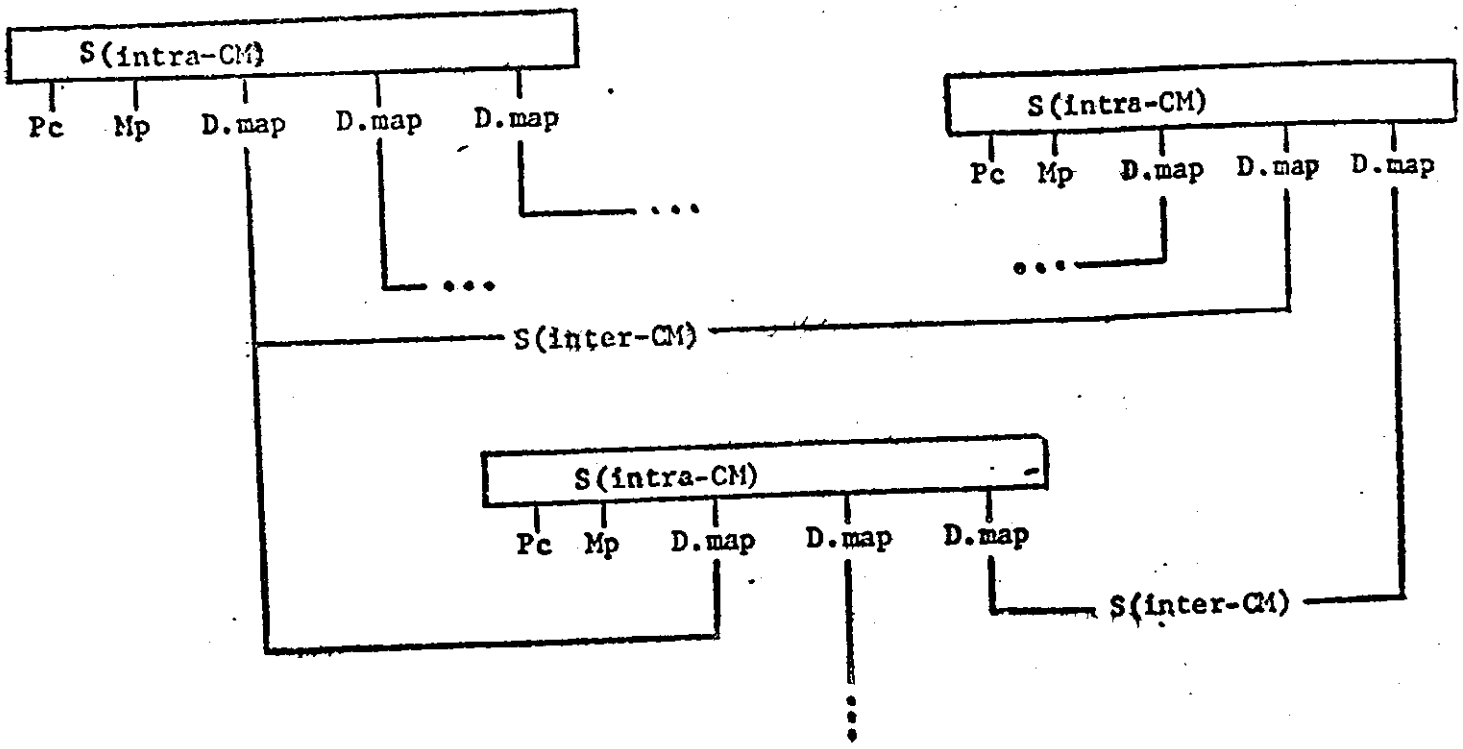


Figure 5. The General Structure of a Computer Module System

Approximately 30% of the CM cost can be attributed to the I/O ports. This is the lower bound found in RT level module sets overhead (30%~70%). Thus 30%~50% appears to be a good estimate of the hardware price to be paid for the convenience of having a higher level module set.

A number of interesting research problems arise from the study of PMS level modules:

1) Capacity of intermodule links

2) Deadlocks

3) Intermodule control mechanisms

4) Process-to-module binding

5) Problem decomposition

The last point is particularly important. Methods for decomposing a problem so that it can be executed in parallel on several interconnected, cooperating PMS level modules must be developed in order to realize the potentiality of the PMS modules.

6 - CONCLUSIONS

PMS level modules could be available in the next 4-6 years. Their existence will open many significant areas of research. It appears that the overhead for PMS modular systems will be on the order of 30%~50% but with decreasing hardware costs this will be tolerable. The expendable components will be processors and there will be no effort to obtain a high utilization factor for the individual processors in a system. An 80%~90% idle time may be acceptable. The high sales volume required by the semiconductor industry suggests that, in the foreseeable future, PMS level components will be oriented towards mass market applications like personal calculators and intelligent terminals. It is interesting to note that as the cost per digital function has decreased the design time and cost per system has remained relatively constant. So instead of obtaining a cheaper system with the same functions a user gets a more complex system at the same cost. This is best exemplified by observing the evolution of minicomputers and noting that the cost per system of a 1965 vintage minicomputer (e.g., PDP-8) is about as costly as a 1974 minicomputer (e.g., the PDP-11)* [Bell, 1974]. Finally, microprogrammed modules are an attractive control element for PMS level modules from both an economic and a transparency point of view.

---

* The complexity of a minicomputer can be divided into a constant and a variable complexity portions. The constant portion of the design have not changed significantly with the technology, e.g., the memory, bus, and I/O structure. The variable parts of the design (registers, ALU, functions, etc) are easier to implement with more advanced technology. Hence more complex functions have appeared for the same cost.

7. REFERENCES

[Barbacci, 1973] Barbacci, M.R. and D.P. Siewiorek: "Automated Exploration of the Design Space for Register Transfer (RT) Systems". Proceedings of the First Annual Symposium on Computer Architecture. University of Florida, Gainesville. ACM SIGARCH, Computer Architecture News, Vol. 2, No. 4, December 1973, pp. 101-106.

[Bell, 1971] Bell, C.G. and A. Newell: "Computer Structures: Readings and Examples". McGraw-Hill, New York, N.Y. 1971.

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

[Bell, 1972b] Bell, C.G., J. Grason, and A. Newell: "Designing Computers and Digital Systems". Digital Press, Digital Equipment Corporation, 1972.

[Bell, 1973] Bell, C.G., R.C. Chen, S.H. Fuller, J. Grason, S. Rege, and D.P. Siewiorek: "The Architecture and applications of Computer Modules: A set of Components for Digital Design". IEEE Computer Society International Conference, CompCon73, March 1973, pp. 177-180.

[Bell, 1974] Bell, C.G. (Private Communication).

[Clark, 1967] 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 Proceedings, Vol. 30, SJCC 1967, pp. 335-402.

[Davidow, 1972] Davidow, W.H.: "General Purpose Microcontrollers. Part 1: Economic Considerations". Computer Design, Vol. 11, No. 7, July 1972, pp. 75-79.

[Freeman, 1971] Freeman, P. and A. Newell: " A Model for Functional Reasoning in Design". Second International Joint Computer Conference on Artificial Intelligence, London, September 1971.

[Fuller, 1973a] Fuller, S.H. and D.P. Siewiorek: "Some Observations on Semiconductor Technology and the Architecture of Large Digital Modules". Report of the Workshop on the Architecture and Applications of Digital Modules held on June 7-8, 1973. IEEE Computer, Vol. 6, No. 10, October 1973, pp. 14-21.

[Fuller, 1973b] Fuller, S.H., D.P. Siewiorek, and R.J. Swan: "Computer Modules: An

Architecture for Large Digital Modules". Proceedings of the First Annual Symposium on Computer Architecture. University of Florida, Gainesville. ACM SIGARCH, Computer Architecture News, Vol. 2, No. 4, December 1973, pp. 231-236.

[Heart, 1973] Heart, F.E., S.M. Ornstein, W.R. Crowther, and W.B. Barker: "A New Minicomputer/Multiprocessor for the ARPA Network". AFIPS Conference Proceedings, Vol. 42, NCC 1973, pp. 529-537.

[INTEL, 1972a] INTEL Corporation: "MCS-4 Microcomputer Set Users Manual". Santa Clara, California, July 1972.

[INTEL, 1972b] INTEL Corporation: "MCS-8 Microcomputer Set Users Manual". Santa Clara, California, November 1972.

[Parnas, 1971] Parnas, D.L.: "On the Criteria to be Used in Decomposing a System Into Modules". Department of Computer Science, Carnegie-Mellon University, August 1971.

[Parnas, 1972] Parnas, D.L. and D.P. Siewiorek: "Use of the Concept of Transparency in the Design of Hierarchically Structured Systems". Department of Computer Science, Carnegie-Mellon University, 1972.

[Patil, 1972] Patil, S.S. and J.B. Dennis: "The Description and Realization of Digital Systems". IEEE Computer Society International Conference, CompCon72, September 1972, pp. 223-226.

[Rege, 1974] Rege, S.L.: "Design to Specification of Modular Data Flow Structures". PhD Thesis, Department of Computer Science, Carnegie-Mellon University, 1974.

[Robinson, 1973] Robinson, D.M.: "Digital Systems Design with Control Modules". IEEE Computer Society International Conference, CompCon73, February 1973, pp. 207-210.

[Wulf, 1972] Wulf, W.A. and C.G. Bell: "C.mmp - A Multi-Mini-Processor". AFIPS Conference Proceedings, Vol. 41, part II, FJCC 1972, pp. 765-777.

BIBLIOGRAPHIC DATA SHEET	1. Report No.	2.	3. Recipient's Accession No.
4. Title and Subtitle SOME OBSERVATIONS ON MODULAR DESIGN TECHNOLOGY AND THE USE OF MICROPROGRAMMING			5. Report Date July, 1974 6.
7. Author(s) D. P. Siewiorek and M. R. Barbacci			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Carnegie-Mellon University Department of Computer Science Pittsburgh, Pennsylvania 15213			10. Project/Task/Work Unit No. 11. Contract/Grant No. GJ32758X
12. Sponsoring Organization Name and Address John R. Lehman Program Director Computer Systems Design National Science Foundation; Washington, D. C. 20550			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts As modules become more complex the advantages and disadvantages of modularity have become more pronounced. The cost of modularity is measured not only in added hardware but also in a loss of flexibility. Functions that are easy to implement at a submodule level may be very difficult, or even impossible, to duplicate at the modular level. We term this a loss of transparency. The added hardware and transparency costs are given for existing module sets and projections made for the next generation of modules. Finally, a microprogrammed implementation of the control portion of existing and projected module sets is shown to be a way to decrease hardware costs and increase transparency.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for public release; distribution unlimited.	UNCLASSIFIED	27
	20. Security Class (This Page) UNCLASSIFIED	22. Price

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

"SOME OBSERVATIONS ON MODULAR DESIGN TECHNOLOGY AND THE USE OF MICROPROGRAMMING"

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

D. P. Siewiorek
M. R. Barbacci

8. CONTRACT OR GRANT NUMBER(s)

F44620-73-C-0074

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pennsylvania 15213

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

12. REPORT DATE

July, 1974

13. NUMBER OF PAGES

27

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

Air Force Office of Scientific Research
1400 Wilson Boulevard
Arlington, Virginia 22209

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

As modules become more complex the advantages and disadvantages of modularity have become more pronounced. The cost of modularity is measured not only in added hardware but also in a loss of flexibility. Functions that are easy to implement at a submodule level may be very difficult, or even impossible, to duplicate at the modular level. We term this a loss of transparency. The added hardware and transparency costs are given for existing module sets and projections made for the next generation of modules. Finally, a microprogrammed implementation of the control portion of existing and projected module sets is shown to be a way to decrease hardware costs and increase transparency.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)