(27)

THE ARCHITECTURE AND APPLICATIONS OF COMPUTER MODULES: A SET OF COMPONENTS FOR DIGITAL SYSTEMS DESIGN*

C. Gordon Bell**, Robert C. Chen, Samuel H. Fuller,
John Grason, Satish Rege and Daniel P. Siewiorek

Departments of Computer Science and Electrical Engineering
Carnegie-Mellon University

ABSTRACT

This paper discusses the design and use of system-building modules of about minicomputer complexity. These modules (CMs), are intended to facilitate the design of the full range of digital systems needed to carry out current and future computational tasks.

INTRODUCTION

Module sets for computer system design are becoming increasingly complex, driven by decreasing cost and size of hardware and increasing computer system performance requirements. Standardized module sets have evolved from circuit elements to gates and flip-flops to IC chips to register transfer level module sets. In this paper we introduce a new, more complex, more flexible set, called Computer Modules (CMs).

A CM consists of a processor (Pc) and memory (Mp) of about minicomputer complexity, together with several carefully designed ports: see Figure 1. The I/O and interrupt structures of conventional computers make it difficult to use them to construct closely coupled networks. Addressing this problem, each port of a CM is designed to handle operations such as handshaking and buffering, executing concurrently with the processor of the CM. These ports allow us to construct CM systems covering a wide range of cost and performance.

CMs will come into physical existence within the next few years: the current microprocessors of Intel, National Semiconductor and AMI [10] are precursors of these modules. This paper presents the rationale for the trend towards more complex modules and for CMs in particular. The range of application of CMs and the implications, i.e., efficient communications interfaces, are discussed. Finally, we describe some applications, and discuss the communications needs and cost/performance tradeoffs related to these applications.

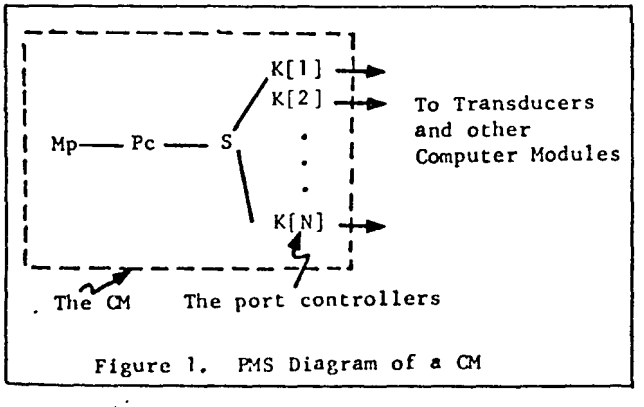

Figure 1. PMS Diagram of a CM

ARCHITECTURE

Module Complexity

CMs are another step in a continuing trend toward more complex modules. One reason for this trend is that defining more complex modules from simpler modules yields distinct advantages. One advantage is that the complex modules can be used in different applications without redesign, and larger systems of these modules can be constructed or modified faster and more easily. A second advantage arises from economics of scale: by standardizing the more complex modules, they can be mass produced rather than made on a custom basis.

The trend toward more complex modules is also due to decreasing size and cost of hardware and increasing pressures for complex systems. The pressures for complex systems come from several sources. One is the need for high performance systems such as the IBM 360/91 [2], CDC STAR [3] and ILLIAC IV [3]. Another is increasing awareness of the advantages of decentralized systems; this is evidenced by the appearance of powerful I/O processors, multiprocessor systems such as C.mmp [14], computer networks such as the ARPA net, and increasingly intelligent terminals.

Increasing module complexity also has disadvantages. These include (a) inflexibility of structure below the module level resulting in both suboptimal use of resources and suboptimal performance, and (b) inflexibility of module function resulting in suboptimal systems design. In order to

*The research in this paper was supported by National Science Foundation Grant GJ 32758X.
**On leave at Digital Equipment Corporation, Maynard, Mass.

1. Communication links: single words and data strings (low speed).
2. Word-by-word inter-computer buffers: single words, program-controlled.
3. Block transfer among computers: data strings.
4. Sharing memory among multiple processors: random access to global variables.
5. Sharing peripheral devices: block transmission (e.g. disk).
6. Broadcast of data/control to multiple, independent units.

Table 1. Common Communication Mechanisms

reduce these disadvantages, any module set consists of a range of module types. At some point, however, the number of module types begins to nullify the advantages that can be gained by standardizing the modules. Register transfer level modules are already bumping against this ceiling. Any set of more complex modules would either take too many module types or become too inflexible in function or form, if it were not for that classic idea: stored programs. Since CMs are programmable, any one module type in the CM module set can perform any function, and it is not necessary to have many different CM types. At this point we envision a CM set where different types of CM differ only in size of primary memory (Mp) and design of I/O ports (e.g. ports for communication between local or remote CMs and ports for interfacing with conventional device controllers). Any one CM, however, can occupy only one point in the cost/performance space. To obtain a variety of levels of cost and performance, CMs are interconnected into systems of varying size and complexity.

The CM can also be viewed as part of the evolution of current centralized computer structures into highly distributed, intelligent networks. This evolutionary sequence has proceeded from (a) single processor with centralized control of I/O, through (b) the addition of interrupts and local control of I/O, (c) I/O processors, and (d) multiple central and I/O processors, finally to (e) multiple, interconnected computers. Almost every current I/O device (e.g. typewriter, card-reader) and secondary storage device (e.g. magnetic tape, disks) could be controlled by CMs in future systems. This not only permits more concurrency as well as autonomy with respect to a central computing site, but also permits better local control for higher reliability and better failure diagnosis.

Communications

The wide range of performance for CM systems, discussed above, will be possible only if CMs can cooperate efficiently in parallel systems. This requires that the modules be able to synchronize and communicate efficiently. In fact, this is the major factor that differentiates CMs from current minicomputers.

Various physical communications structures as well as software protocols have been proposed or built to satisfy the needs of various parallel systems. One instance is the highly multiplexed switch of the C.mmp [14]. Another is the asynchronous, extendable ring of the Distributed Computing System [8]. Still others are the geographically dispersed ALOHA [1] and ARPA network [12] systems, the bus systems such as the PDP-11 Unibus [6], and the synchronized, highly structured ILLIAC IV system.

The more common communication mechanisms used in such systems are given in Table 1. These range from the conventional communications link for pair dialogues to the shared primary memory which permits any processor to access any global variable. The various dimensions of the interconnection problem are (a) logical switching structure (none: single transmitter-single receiver; broadcast: single transmitter-multiple receiver; Unibus-type: single pair dialogues broadcasted), (b) physical switching structure (links + central switch; bus; loop), (c) node separation (local; distributed), (d) message type (1-bit events; data word; variable name + data word; data blocks), and (e) node addresses (none; single address; subset of nodes).

This wide variety of systems is representative of the flexibility required of CM interfaces. This indicates that CMs must have several communications ports, each with facilities for handshaking and other forms of synchronization.

Efficiency often requires that module interfaces have independent processing power. For instance, in many cases independently controlled buffers should be provided that do not have to be directly managed by the central processor. Interrupt queueing and some simple interrupt processing might also be performed independently of the central processor. CM ports must therefore have sufficiant power and flexibility to construct efficient parallel systems.

APPLICATIONS

In order to learn more about how CMs should be designed, and also to illustrate how they might be used to design systems, we have investigated a set of applications, including array processing (Fast Fourier Transform processing, generalized array processing, and radar signal processing), sorting, language processing (compilation and machine language interpretation), and process control. In each case, we have tried to bring out the communications requirements and the range of performance that can be achieved by varying the CM system structure. In this paper we describe in detail only the sorting and Fast Fourier Transform applications. The other applications are discussed briefly.

Fast Fourier Transform

The Fast Fourier Transform (FFT) lends itself to parallel and pipelined processing [4,5]. For an n-point transform, the algorithm can be represented by an array of nodes with n rows and $\log_2 n$ columns. At each node, two values calculated by the nodes in the previous column are processed, producing a value to

be used in the next column. The calculations of each column must be completed before the calculations of the next column can proceed. Within any column, however, the calculations at each node can proceed independently of the calculations at the other nodes.

For an n-point transform, therefore, four simple CM implementations are: (a) a full parallel/pipeline structure using n log n CMs, one for each of the nodes, (b) a parallel structure using n CMs, one for each of the nodes in a column, in effect folding the columns back on themselves, using an appropriate communications net, (c) a pipeline structure using log n CMs, one for each column, and (d) a completely serial implementation using just one CM. The speeds in each case are approximately proportional to the number of CMs. Furthermore, any node can be implemented using several CMs for higher speed. In this way, different performance requirements can be satisfied by consructing differently structured CM systems.

The communications requirements are different for each of the four implementations discussed. Private or shared (bus or loop) connections, size of buffers required, handshaking protocols and interconnection patterns are some of the considerations.

Sorting

Various sorting algorithms can be implemented by CM systems. Some bucket sorts are chosen for illustration.

Suppose the file to be sorted is divided among n CMs, all connected via a ring-type communication net. Each of the n CMs are pre-assigned a range of values. The CM examines its portion of the file and any record which does not fall within its assigned range is placed on the ring. Records on the ring circulate until picked up by the CM in whose range the record key falls. After all the records have been picked up, each CM will sort the records it collected (this can be done either internally to the CM or by reassigning ranges to the CMs and repeating the distribution process).

Another possibility is to use a tree structure (Figure 2). Here, each record arriving at a node is sent to the upper or lower successor node as appropriate, until it arrives at a leaf node. Each leaf node then sorts the records it collected. This tree indicates a parallel/pipeline structure which can be

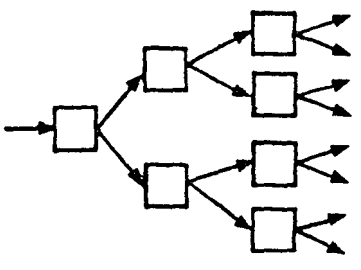

Figure 2. Sorting Tree

mapped in various ways onto CM systems: we could use one CM for each node in the tree, or one for each level, for instance. One might notice that there is a speed mismatch in the ratio 2:1 between any two successive levels of the tree; to correct this, we can use an array structure similar to that for the Fast Fourier Transform. It is important to note that while these structures are similar, the traffic pattern is different and consequently different communications structures may be indicated.

Other Applications

In generalized array processing (such as performed by the ILLIAC IV [3]) and specialized array processing (such as radar signal processing), the outstanding feature in many cases is close coupling of the component CMs. Communication bandwidth requirements are likely to be high, so the CMs may be configured as an array, a vector, or ring, or some other interconnection structure may be used (e.g., the perfect shuffle [13]). Each CM can repeatedly apply one of a series of computation steps, pipeline fashion (as in the CDC Star), or each CM may apply all the steps to one particular data item at a time (as is typical for the ILLIAC IV).

In compilation, partitioning the compilation procedure into phases allows pipelining of compiled programs, while techniques similar to incremental compiling [7] can be applied to obtain parallelism at the subprogram level. The outstanding feature in the communications structure is the common data base (symbol table, etc.) accessed by most of the CMs at each stage of the compilation. This data structure may be stored in one memory (which must field requests from any of the CMs) or may be duplicated at each CM. These different schemes imply different communications structures.

In machine language interpretation, parallelism is obtained by pipelining instruction fetch, decode and execution, and by multiplying the number of execution units. The demands made on the communications system involves broadcasting of to-be-executed instructions, buffers at the execution units, etc. A fine example of a high-performance structure is the 360/91 [2]. Better system utilization (and therefore better performance) can be achieved if the CM system is used to execute simultaneously more than one program, or interpret parallel machine languages [11].

In the case of process control computing, the computation can usually be done in parallel in terms of multiple independent control tasks. A system of this type is typical of (a) telephone switching (e.g. control of switching paths in terms of dialed route requests), (b) discrete process control (e.g. a transfer machine requiring simultaneous solution of 800 boolean equations on a time-sampled basis), and (c) closed-loop time-sampled control (e.g. simultaneous solution of multiple independent control equations for various loops). In each case, while many operations are done in parallel, each operation is small, and independent of other operations. Communications among the various control tasks is provided by multiple links to process inputs and outputs and to other computers (See Figure

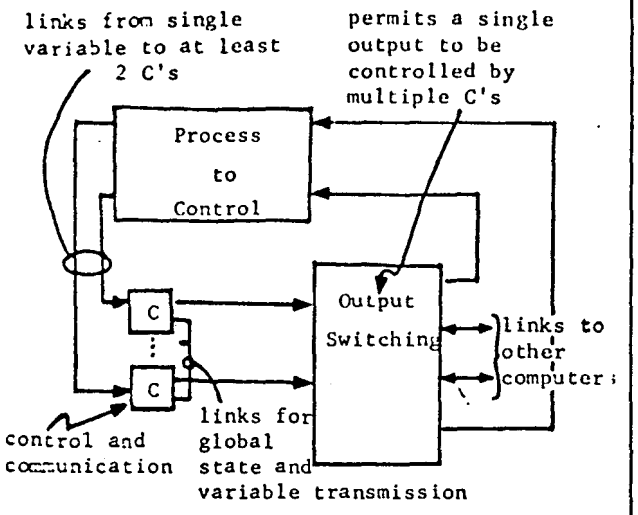

Figure 3. Process control for a set of computers permitting redundant control paths.

3). Each CM also requires communications to all other CMs in order to access common input variables, communicate global state variables, and to report status. The communication requirements are similar to those encountered when using multiple CMs for compilation or interpretation.

CONCLUSION

We have discussed the architecture of CMs and the possibility of constructing computing systems with them. Although several questions remain to be answered about the CM's architecture, several preliminary conclusions emerged from this study: (a) a microprocessor is included within each CM (b) the structure of the I/O ports is crucial, and (c) more than one I/O port is needed per CM. Table 2 gives some of the characteristics we expect CM systems to have, based on the applications we investigated.

Major questions remain. Can CM structures successfully compete with conventional computers? (We do not necessarily expect them to execute machine language faster than an emulated machine, but, given a set of applications, we wonder how well a CM system would do compared to a conventional computer.) How well will a CM structure fit a set of applications? Is there a general structure suitable for most applications? (Some work has been done in this direction [11].) How can we design for a given reliability requirement? How do we specify the communication requirements for a given application, and how do we design physical communication structures and inter-module protocols to fit these? These and other questions must be answered. They will be subjects of further research.

Attribute	Values
No. of processors	1
Memory size	1K words and over
Word size	8 to 16 bits
No. of ports	2 to 5
No. of CM types	1 to ?
No. of CMs in a system	A few to several thousand

Table 2. Properties of CMs and CM Systems

REFERENCES

1. Abramson, N., "The ALOHA System - Another Alternative for Computer Communications," FJCC Proceedings 1970, Vol. 37, pp. 281-285.
2. Anderson, D. W., Sparacio, F. J., Tomasulo, R.M., "The IBM System/ 360 Model 91: Machine Philosophy and Instruction-Handling," IBM Journal of Research and Development, Vol. 11, pp. 8-24, January, 1967.
3. Barnes et al., "The Illiac IV Computer," IEEE Trans. Computers, Vol. C-17, No. 8, pp. 746-757, August 1968.
4. Bergland, G. P., "Fast Fourier Transform Hardware Implementations--An Overview", IEEE Trans. on Audio and Electroacoustics, Vol. AU-17, pp. 104-119, June 2, 1969.
5. Brigham, E. O. and Morrow, R. E., "The Fast Fourier Transform," IEEE Spectrum, Vol. 4, pp. 63-70, December 1967.
6. Digital Equipment Corporation, "PDP-11 Interface Manual," DEC-11-HIAB-D, Digital Equipment Corporation, Maynard, Mass., 1971.
7. Earley, J. and Caizergues, P. "A Method for Incrementally Compiling Langues with Nested Statement Structure", Comm. of the ACM, Vol. 15, pp. 1040-1044, December 1972.
8. Farber, D. J. and Larson, K. C., "The System Architecture of the Distributed Computer System - The Communications System," Symposium on Computer Networks, The Polytechnic Institute of Brooklyn, April, 1972.
9. Hintz, R. G. and Tate, D. P., "Control Data STAR-100 Processor Design," IEEE Computer Society International Conference, September 1972.
10. Lapidus, G., "MOS/LSI Launches the Low-Cost Processor," IEEE Spectrum, Vol. 9, pp. 33-40, November 1972.
11. Lesser, V. R., "Dynamic Control Structures and their use in Emulation", Ph.D. Thesis, Computer Science Dept., Stanford University, August 1972.
12. Roberts, L. G., Wessler, B. D., "Computer Network Development to Achieve Resource Sharing," SJCC Proceedings 1970, Vol. 36, pp. 543-549.
13. Stone, H. S., "Parallel Processing with the Perfect Shuffle," IEEE Trans. Computers, Vol. C-20, pp. 153-160, February 1971.
14. Wulf, W. A., Bell, C. G., "C.mmp - A Multi-Mini-Processor," FJCC Proceedings 1972, Vol. 41, Part II, pp. 765-777.