Microelectronics and Computer Science*

by

Arvind
Kim P. Gostelow

Technical Report #106

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

*This work was supported by NSF grant MCS 76-12460.
UCI Dataflow Architecture Project.

Microelectronics and Computer Science*+o

by

Arvind

and

Kim P. Gostelow

Department of Information and Computer Science
University of Caliornia, Irvine
Irvine, California 92717

ABSTRACT

Technology has a profound influence on computer science. In the past, technology has always been a constraint and has required computer designers to organize computers around what was feasible rather than what was desired. Whenever new technology arrived (e.g., transistors in place of tubes) the line of feasibility simply moved. However, microelectronics has brought far more severe changes and the line of feasibility has not only moved drastically, even moving beyond what computer scientists were prepared for, but is now requiring us to reconsider the basic conceptual precepts upon which (almost) all computers have rested. That is, computer scientists have been confined totally to the so-called von Neumann class of computers and their three basic attributes: low-level machine language, centralized sequential control, and linear memory organization. Machines based upon these characteristics have worked fairly well in the past (the last 30 years), but recent developments in LSI technology have changed these attributes themselves into inappropriate constraints. For example, LSI favors distributed and asynchronous computer operation -- a principle incompatible with von Neumann machines. Also, since basic components are far more complex, and thorough testing is economically unfeasable, achieving reliability must be a primary design consideration.

---

*This work was supported by NSF grant MCS76-12460
+This paper was presented at the 2nd IEEE(G-PHP)/ISHM University/Industry/Government Microelectronic Symposium at the University of New Mexico, Albuquerque, New Mexico, January 3-5, 1977.
°Revised with minor editorial changes: 15 August 1977.

1. Introduction

Many people in the computer science field speak with great fascination about the possibility of placing a computer system such as an IBM 370 into a small box within the next few years. Such a system would certainly be a change from the equipment currently required. However, the statement also indicates that the designers of microprocessors and microelectronic devices essentially work with the same concepts of computer architecture that were the basis of almost all previous computers. The net result of this trend is that while computers themselves have become substantially less expensive, the cost of using computers has, if anything, increased. The techniques for producing applications software (which constitutes the major part of the cost) have not improved sufficiently to meet the rising expectations and demands.

Undoubtedly the impact of miniaturization of computers is far-reaching. Miniaturization has permitted the use of computers in space vehicles, thereby significantly increasing the scope of on-board experiments. Reduction in both size and cost of computers has potentially opened such a large area for the applications of computers that far reaching social implications will necessarily follow. However, in spite of a wide range of applications,

microelectronics has not increased our ability to solve complex computational problems.

A precise (and common) measure of the complexity of a problem is the time and space consumed by a machine executing a programmed solution to the problem. Many problems in astrophysics, weather prediction, and game playing are examples of very complex problems. Problems such as these cannot be effectively sped-up simply by reducing gate delays on a chip, even by a factor of 10 or 100 (if such were physically possible). Rather, increases in machine performance must come from new and very different internal logical orgnizations for computers. However, past efforts in developing new computer architectures, as exemplified by ILLIAC IV, have had only modest success. Computer organizations such as ILLIAC IV unfortunately require much greater programming effort and necessarily make software related problems even more difficult to solve. Solutions such as ILLIAC IV are not appropriate, and we hope to show why this is so.

There is another kind of complexity that relates more to the structure of computation itself. Consider the problem of designing an operating system. It is very difficult to view an operating system as a single sequence of instructions. A more natural, and less complex view, is that of several independent resource manager programs occasionally interacting with one another. When the basic

structure of a problem solution exhibits this non-sequential behavior, it becomes very difficult to program the solution on a computer that is based on a single sequence of instructions. Many problems in modelling, especially in economics, also fall into this category. Simple miniaturization or reduction of the hardware cost of computers will have little effect on solving this type of complex problem.

The remainder of this paper shows why microelectronic researchers and developers on the one hand, and computer scientists on the other, must begin to exchange ideas in order to solve complex problems in computing. Computer scientists must develop new principles for organizing computation which have not existed previously and which are reasonable in the eyes of microelectronic researchers. Conversely, microelectronics developers must realize the fundamental limitations of current computer architectures and understand what new principles are to be applied in order to produce machines which will be useful.

This paper is organized as follows: Section 2 shows how technology has influenced computer science in the past and why the impact of LSI on computer science is at a very fundamental level. Section 3 discusses three major problems related to the utilization of microelectronics, and Section 4 briefly outlines one approach to computer design which advocates new principles to effectively utilize

microelectronics to solve complex problems. Our conclusions appear in Section 5.

2. The Influence of Technology on Computer Science

The feasibility of many ideas in computer science depends upon the technology available at the time the ideas are conceived. Many ideas are discarded outright because they cannot be realized economically, as for example a large associative memory. The effects of technology on computer science can be more subtle when designers see the technology placing firm constraints on them. Consider memory systems: designers have always worked under the assumption that fast memory is expensive. Consequently all machines have hierarchically organized memory systems.

The effect of such constraints on computer science has been to devise techniques and computer organizations to overcome and reduce the limitations imposed by technology. Virtual memory, pipelining, overlapped I/O and central processor operations, and microprogramming are all examples of successful solutions to technological constraints. These particular solutions have, in turn, increased our understanding of the basic principles of machine organization. However many other technological constraints have caused effort to be expended in problems of dubious value, often in the area of efficiency studies and optimization.

The influence of technological constraints can even live beyond the technology that brought them. The LSI version of the PDP-11 is an example of such constraints. What previously was an entire processor is now available as a component. The constraint being applied (unconsciously) is that the PDP-11 is the end product, when in fact, we wish to be able to use it as just one processor within an ensemble of many processors. As evidence, consider a box full of PDP-11s. We expect intuitively that under some proper bussing structure adding more processors to that box should give us greater computing "power", but so far it has not. In fact, we conclude that it cannot because the PDP-11 is not a module, and there is no suitable interconnection to make it so. None of the microprocessors available today can be effectively interconnected in order to form a more powerful computer. The main source of difficulty is the programmability of such configurations rather than the hardware interconnection problems.

Finally, if we take an historical view, we can see that previous changes in technology have been readily absorbed by computer designers. Tubes were replaced by transistors, hardwired controls by microprograms, etc. without traumatic effects on the users. The advent of microelectronics has posed far more severe changes, the effects of which actually reach the very foundations of computer science.

3. A reevaluation of von Neumann-type computers

Von Neumann-type computers are characterized by the following three precepts:

1. Low-level machine language (commands are elemental operations on elemental operands).
2. Sequential centralized control of the computational process (control is a single sequence of commands, each executing one operation and passing control to the next command, as determined by a simple cell called the instruction counter).
3. Linear organization of storage consisting of identical sequentially-numbered cells.

It is important to recognize that these principles were developed before 1950 directly as a result of the technological constraints of that time, and even though new technology has now removed many of those constraints, the principles are still with us. In fact, it is fair to say that the views of mainstream computer scientists regarding computer system organization are not just biased by, but are almost totally confined to, von Neumann-type architecture.

3.1 The difficulty in utilizing microelectronics

Von Neumann's principles worked well in the past and have withstood considerable change in technology. But now these principles are hindering the effective utilization of new technology [1]. In fact, this should come as no surprise since microelectronics is basically disposed towards systems that are distributed and comprise large numbers of similar components. But computers (or modules)

constructed on von Neumann's principles simply cannot cooperate effectively with one another towards a common goal. A single sequential instruction stream and the low level of operations mitigate against any such mutual cooperation.

Von Neumann himself recognized that a very complex organization cannot function effectively with a centralized sequential control [2] -- the second principle above. He suggested alternative organizations of automata, and one such organization was based on a large number of identical and asynchronously cooperating cells (processing elements). (In fact, tne first computer, ENIAC, designed by Mauchley and Eckert, possessed no central control and did operate with asynchronous processing units.) These cells were not controlled by a central sequential command scheme, but based their operation on a small program internal to each cell and observation of the state of their immediate neighbor cells. It was this scheme which von Neumann used to devise his self-reproducing machine (a decidedly theoretical investigation). Holland [3], actually proposed a machine based on these ideas in 1959. The most significant fact about Holland's machine was that the technology was inadequate to warrant further consideration. Not only could such cells be economically produced in large numbers now, but far more sophisticated building blocks could be devised to serve as the basis of distributed and asynchronous machines.

3.2 The high cost of software

Software now accounts for the majority of costs in a computer system. Microelectronics will only make this fact more pronounced by reducing hardware costs further. The observation that machine architecture contributes to the high cost of software is not appreciated by many. This view stems primarily from the fact that virtually all computers and languages have the same old central sequential control structure as their fundamental basis of operation (including ILLIAC IV and APL). Only when programming with sequential control becomes absurd are other control structures considered. For example, special programming tools are designed to provide systems programmers with an asynchronous or non-sequential control facility to program operating systems. Since asynchrony is not a fundamental mode of operation of von-Neumann type machines, the execution effeciency of these tools is low. Also, available tools are simplistic in nature and are totally inadequate to provide a viable asynchronous programming base. Furthermore, the influence of an ever-present sequential machine on a programmer's thinking and on the systems that result must be considered. These costs are difficult to assess, but we believe that a computer with asynchronous operation at the machine language level will contribute significantly to reducing the cost of software. This is particularly important when we recognize that the hardware and software systems currently being produced are miniscule relative to

those which will be produced in the future. Lastly, it is important to note that such a machine cannot be obtained by simply "adding asynchrony" to a von Neumann computer.

The above statements have concentrated on von Nuemann's second principle (asynchrony) because we consider it the most important and least understood of the three. The remaining two principles (low-level machine language, and linearly organized memory) are also liable to reexamination, and it can be argued, only contribute to software costs.

3.3 Reliability

Since the cost of manufacturing LSI components is already less than the cost of testing those components, one must accept machines built from parts which are unreliable. Problems of fault tolerance, error detection, self-repair, etc., must now be addressed directly. Computer scientists, as well as microelectronics researchers, will have to consider these aspects of reliability as basic and consider them explicitly in the design of components and computers.

4. An Alternative Computer Organization

The preceeding sections have presented arguments to demonstrate that very different computer architectures are needed in order to effectively utilize microelectronics to solve very complex problems. One new principle of machine operation which may answer the need is called dataflow.

Computers based on the dataflow concept are being developed at the University of California, Irvine [4,5] and at the Massachusetts Institute of Technology [7].

The major deviation from von Neumann's principles is that dataflow is highly asynchronous. Programming in dataflow is done under the illusion that as many processors as needed are available. In the following, we will give a brief outline of what dataflow means, and we will give an idea of the kinds of demands a dataflow computer would place on microelectronics for its implementation.

4.1 Dataflow

By dataflow, we mean a machine language for expressing computation in which

1. An instruction executes when and only when all operands needed for that instruction become available, and
2. Instructions, at whatever level they might exist, are purely functional and produce no side-effects.

In essence, a dataflow program is a partially ordered set of instructions to be carried out, with sequencing implicitly present only as needed for the production of partial results. Data flow languages have been developed [6,8] which include the expressive power of conventional languages with looping, recursion, and conditional constructs. Reference [4], however, gives an entirely new

mechanism for executing dataflow programs which brings out even more asynchrony than previously possible. This execution mechanism, for example, can automatically "unravel" loops during program execution to allow an asynchronous machine composed of many processors to execute all iterations of that loop with as much parallel activity as possible. These parallel or asynchronous activities could include simultaneous calls on complex operations such as subroutines, etc.. The primary resulting effect is to allow the executing hardware to exchange blocks of processors for slices of time, and to allow a machine to execute a program faster as more processors are provided to it. Some details as to how this is done are covered in [4]; we state here only that no preprocessing of programs is required, and that the asynchrony inherent in dataflow programs is brought out by the machine in a very simple and mechanical way.

4.2 A view of a dataflow architecture

In this section we wish to give an idea of what a particular dataflow machine [5] may require in terms of microelectronics. The numbers given are very preliminary and are meant only to convey a general impression of our design at this time.

A basic view of the architecture is shown in Figure 1.

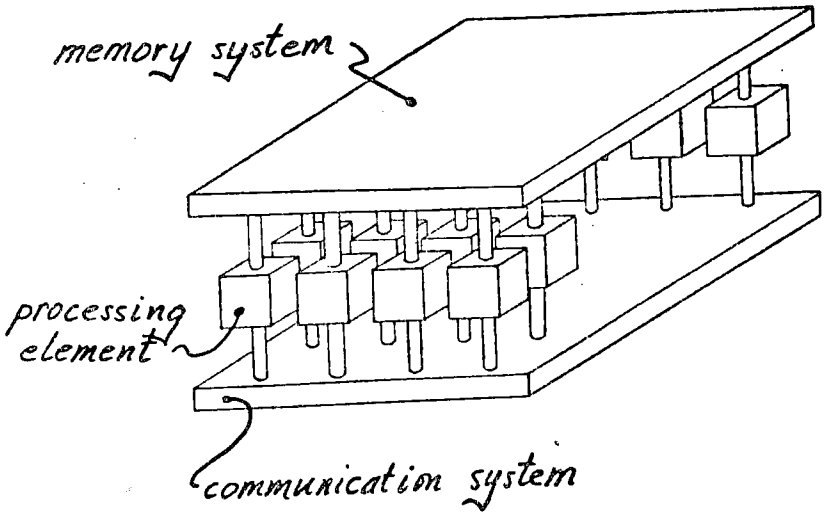

Figure 1

A basic view of a dataflow architecture

The processing elements (PEs) are grouped in columns, each column being composed of 10-20 processors. The complexity of each PE is of the same order as an Intel 8080. During machine operation, a processing element assumes responsibility for accepting input operands along with a specification of what to do with those operands. After accepting the operands, the processor will compute a result and send that result on to some other processor. Which processor will accept which operands is not pre-specified, but is decided completely during program execution.

A processor outputs operands for input to other processors by constructing data packets of 40-100 bits length per datum per destination processor. Each such packet traverses a portion of the communication system in an attempt to find an empty processor or that processor which is waiting to accept that packet. A view of the communication system is shown in Figure 2.

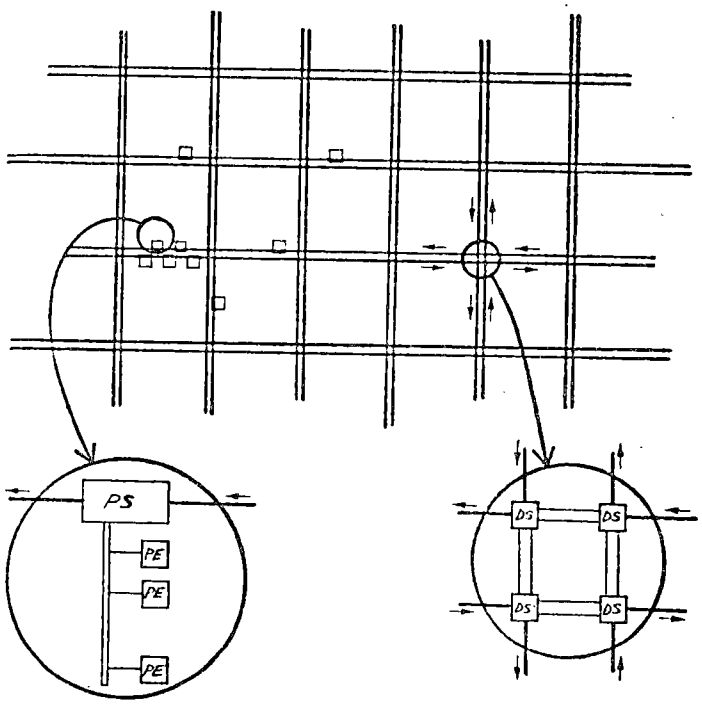

Figure 2

The communication system

The communication system is essentially a set of intersecting bidirectional buses along which operand packets flow. Each processor column is attached to a bus. To reduce the number of columns that a packet must search in order to find its destination processor, the communication system partitions itself into domains by setting domain switches (DSs in Figure 2) at bus junctions to confine traffic to a given domain. A domain corresponds roughly to a subroutine in execution. These switches must be set automatically as subroutines are called, and as subroutines are exited the domains must be deleted and made available for reallocation to (parts of) new domains.

Preliminary calculations indicate that a bus should be capable of supporting a rate of 10-20 million bits per second. The bus must also supply the memory necessary to hold a packet while the packet searches for a destination processor. This suggests that shift registers be used to implement the bus system. Also, the processor/bus junction points must be capable of matching dynamically generated processor names against packet destination names as the tokens pass by each column. This matching mechanism implements the processor search (PS in Figure 2) which each packet must do.

Lastly, we mention the operand memory where some of the (generally) tree-structured data values may be kept. The memory is not seen by the programmer, and is present only in

order to reduce the amount of data that the communication system would otherwise be required to carry. The memory is distributed over the entire machine, yet each processor must be able to access any piece of data in the memory. The farther the data is located from the requesting processor, the longer the time to retrieve that data. A novel memory busing system which may be useful for handling the data referencing characteristics we expect is discussed in [9].

To summarize the system, it is asyncrhonous and without centralized control; it is also highly modular and new processors can be added at any point. The machine's asynchrony thus departs from von Neumann's principle of a centralized sequential control, and in this respect the machine attempts to capitalize on the character of microelectronics.

5. Conclusions

We have argued in this paper that in order to effectively utilize microelectronics a radical departure from the principles of von Neumann-type computers is needed. The design of future machines must incorporate the following characteristics:

1. The default operation of the base machine must be asynchronous.
2. The design must compensate for unreliable components.

3. The basic increase in the speed of computation must come from the organization of the machine rather than the raw speed of the components.

Without the incorporation of these ideas, miniaturization and the reduced cost of hardware is not going to significantly increase our ability to solve complex problems.

ACKNOWLEDGEMENT

Thanks to Shirley Rasmussen for typing this paper, and to Dave Farber for reading of an early draft.

REFERENCES

[1] Glusnkov, V. M., M. B. Ignatyev, V. A. Myasnikov and V. A. Torgashev, "Recursive Machines and Computing Technology" Information Processing 74, North-Holland Publishing Company, Stockholm, August 1974, (pp. 65-70).

[2] von Neumann, J. Theory of Self-Reproducing Automata ed. by A. W. Burks, University of Illinois Press, 1966.

[3] Holland, J. C., "Iterative Circuit Computers" Proceedings Western Joint Computer Conference, 1960, (pp. 259-265).

[4] Arvind and K. P. Gostelow, A New Interpreter for Data Flow Schemas and Its Implication for Computer Architecture TR72, Department of Information and Computer Science, University of California, Irvine, November 1975.

[5] Arvind and K. P. Gostelow, A Computer Capable of Exchanging Processing Elements for Time, TR77, Department of Information and Computer Science, University of Caliornia, Irvine, January 1976.

[6] Arvind, K. P. Gosteow, and W. Plouffe, Programming in a Viable Data Flow Language TR89, Department of Information and Computer Science, University of California, Irvine.

[7] Dennis, J. B., D. P. Misunas, "A Preliminary Architecture for a Basic Data Flow Processor", The 2nd Annual Symposium on Computer Architecture, Houston, January 1975, ACM-SIGARCH vol 3, No. 4, Dec. 74 (pp. 126-132).

[8] Weng, Kung-Song, Stream-Oriented Computation in Recursive Data Flow Schemas M. S. Thesis (MAC Technical Memorandum 68), Department of Electrical Engineering and Computer Science, MIT, October 1975.

[9] Wittie, L. D. "Efficient Message Routing in Mega-Micro-Computer Networks" Proceedings 3rd Annual ACMSIGARCH-IEEE Symposium on Computer Architecture, 19-21, January 1976, (pp. 136-140).