LEVEL II

ADA081346

DTIC
ELECTE
MAR 6 1980
S D
C

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
1807-1856-1920

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

[illegible] COPY

79 12 19 060

1

14 TR-825 ✓ 11 Nov [redacted] 79

12 28 6

ZMOB: A Mob of 256 Cooperative Z80A-Based Microcomputers

10 Chuck Rieger
Computer Science Department
University of Maryland
College Park, MD 20742

9 Technical rept.

The research described in this report was funded in part by the Office of Naval Research under grant number N00014-76C-0477, and in part by the National Aeronautics and Space Administration under grant number NSG-7253.

15 N00014-76-C-0477, NSG-7253

409022

ZMOB: A Mob of 256 Cooperative Z80A-Based Microcomputers

Chuck Rieger
Computer Science Department
University of Maryland
College Park, MD 20742

1. Introduction

Current directions of computer science and computing in general are toward more parallel machine architectures and distributed models of computing based upon these new architectures. Broadly speaking, parallel architectures fall into two categories: parallel synchronous machines, which execute the same code lock-step in many processors operating on different data, and parallel autonomous machines, in which many independent processors can be put to work on different aspects of a large computation.

In past and much current work, emphasis has been on the former variety of machine [1-3]. Recently, ~~however~~, there has been considerable interest in highly parallel architectures capable of supporting complex distributed computation via a large number of autonomous processors, [4,5]. While many interesting ↳ next page machines have been proposed or are currently being developed, there has apparently been no specific attempt to build a machine with a truly large number of autonomous processors, each having substantial independent computing power.

ZMOB is such a machine, currently under design and simulation, ~~at Maryland~~[a]. Architecturally, ZMOB is a collection of 256 identical but autonomous Z80A-based microcomputers (processors). Each processor ~~(Fig. 1)~~[a] comprises 32K bytes of 375 ns read/write central memory (expandable to 48K bytes), up to 4K bytes of resident operating system on 450ns EPROM, an 8-bit hardware multiplier, and interface logic for communications functions. (This is a non-trivial microcomputer, comparable in size and power to the average small business or personal computer.) Each processor will reside on its own PC board, making a total of 256 processor boards mounted in four rack cabinets, and supplied by a distributed system of local switched power supplies. Although the machine will initially consist of 256 processors, its architecture is extensible (in principle) to any number of processors. In practice, we will anticipate extensibility to 1024 processors. The current cost estimate for the 256 processor machine is $100K.

Good intercommunication pathways and bandwidths are critical to the success of any highly parallel machine. As described below, we have what we feel is a attractive solution to inter-processor communication, and processor-to-outside-world communication. The strategy will be non-blocking (e.g., there can be 128 full-speed conversations between pairs or processors), and will give each processor the illusion of data communication

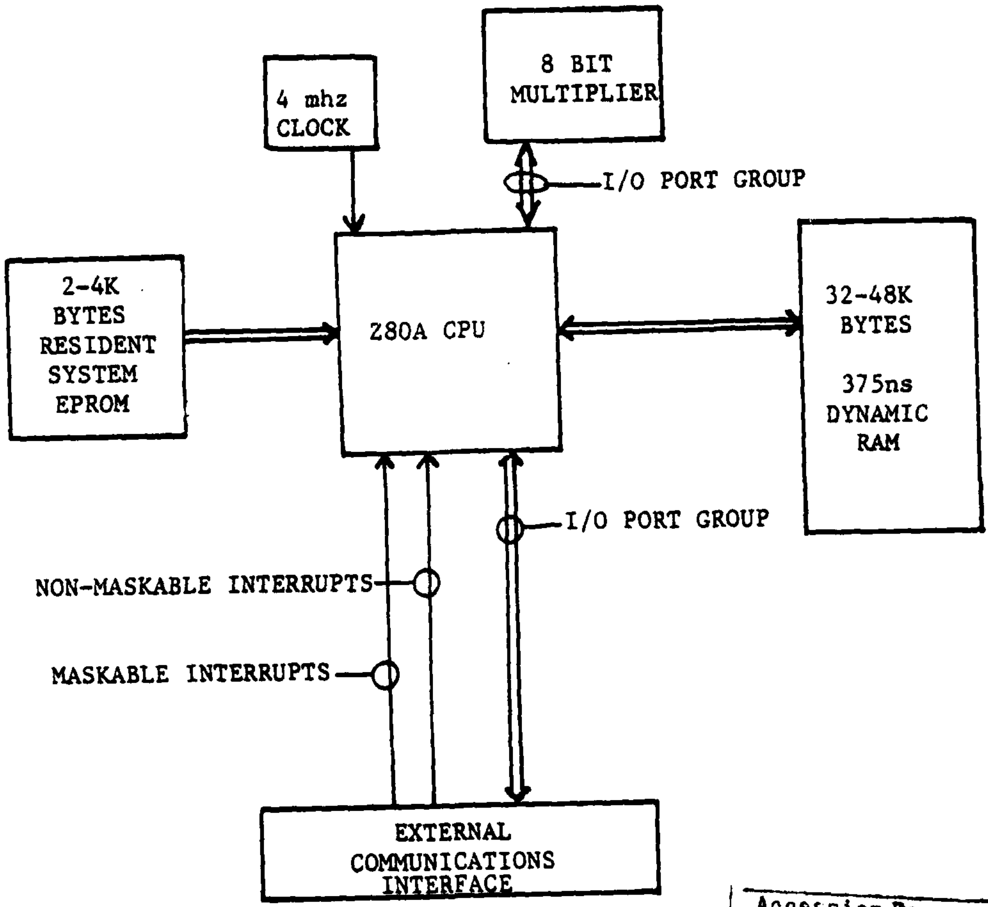

Figure 1. Processor Organization.

rates as high as the Z80A itself can manage. In one mode, for example, data rates into or out of ZMOB can exceed 20 megabytes per second.

2. Background and Motivation

The ZMOB idea sprang originally from needs of the Computer Vision Lab at Maryland. In certain vision tasks based on relaxation algorithms [6], each pixel of, say, a 512 by 512 image must be processed once per "iteration". The processing of each of the quarter million pixels is identical, and independent of the processing of other points during one iteration. A typical relaxation algoritnm will require 5-10 iterations to converge. The particular algorithm used will vary with the application.

In a complex relaxation algorithm, each pixel is represented by a "probability vector" of perhaps 10 bytes, and the per-point, per-iteration computation in such an algorithm might involve 1000 8-bit integer multiplies and a corresponding number of additions. Rough calculations show that one complete iteration can therefore require upwards of 250,000,000 multiplies and a comparable number of adds, a staggering computation which requires hours on a medium-size conventional machine. Our preliminary studies (relatively complete Z80A code, hand-simulated timing results) indicate that ZMOB will require on the order of 100 seconds for such a computation.

Although motivated by relaxation processing needs, it quickly became obvious that a machine with such a great computing potential ought to be general-purpose as a research tool for distributed computing models in all of computer science. Specifically, it became of concern that the geometry of intercommunication paths not be unduly biased by the machine's applications in vision (where 4-neighbor adjacency is natural), and that the 8 megabytes of high-speed central memory not be inaccessible to computations desiring to view the machine as a large single address space. The design of the intercommunication system reflects these concerns, and results in a machine which is both general-purpose, and which (from timing studies) supports the initial vision applications as fast as any other special-purpose Z8ØA-based architecture could.

3. Basic Processor Architecture

ZMOB is a collection of automomous Z8ØA-based microcomputers. The Z8ØA is an 8 bit microprocessor chip with a 158 instruction repertoire, two sets of 7 8-bit registers, and several 16 bit registers for stack pointer, program counter, and indexed addressing. It is a stack machine with a 64K byte address space, 256 logical I/O ports, and a T-cycle time of 25Øns. A typical instruction will require about 2 microseconds to execute, and there are several rather powerful block search and transfer instructions. High-speed vectored interrupt linkage is another virtue of the chip, which sells in quantities for less than $15.

In the initial conception, the plan was to assign each ZMOB processor to two scan lines of image data in a 512 by 512 pixel image. In such a machine, each processor would be connected to its two adjacent processors (handling adjacent scan lines), and to an external controlling computer (e.g., a PDP-11), all over interrupt driven 8-bit parallel ports with handshaking. At power-on, ZMOB would be cleared, bringing each processor back to its basic resident operating system. This system would allow for the loading of applications programs and parameters into the processor's RAM. After initialization, all processors would be forced into their DMA condition while the external machine, having access to the individual processors' address spaces (as pieces of one large virtual space), loaded in the starting image data. After loading and removal of the DMA condition, the external machine would broadcast a system-wide start command over all control ports. Once running, each processor would request information from its neighboring processors, compute two pixels' worth of probability vector updates, then advance to the next of the 512 pixels across its two scan lines. The operation would progress (pretty much synchronously) in all processors simultaneously until, at the end of the scan line, each would broadcast "ready" messages to the external computer. When all had been accounted for, the external computer would again force all processors to their DMA state, read the iteration's results for TV display update, then release the processors on the next iteration.

Timing simulations showed that such a machine would be quite profitable. For example, the time required to compute each image pixel's 3 by 3 8-bit gray-scale average in a 512 by 512 image would be about 2 fifths of a second, while the time required to run a simple edge detector over a 512 by 512 image would be about one second. Even an elaborate relaxation algorithm involving 10 labels per image point would complete each iteration in about 100 seconds, orders of magnitude faster than presently possible on a conventional machine.

The initial conception of the machine rapidly evolved into a design capable of supporting a variety of distributed computing models, in addition to the vision models from which the idea came. The current design, ZMOB, supports the original vision applications within the same time estimates, and will result in a machine that is a flexible and general purpose research tool.

4. <u>The Conveyor Belt</u>

In the current design, there are no direct neighbor-neighbor communication paths. Rather, each processor is a mail stop on a high-speed, synchronous "conveyor belt" (Fig. 2). The ZMOB portion of the conveyor belt is thus 256 positions long (but indefinitely extendible), and resembles certain existing synchronous ring networks in its concept [7] (although the types of messages passed are quite different). The external controlling computer, and perhaps other devices such as high

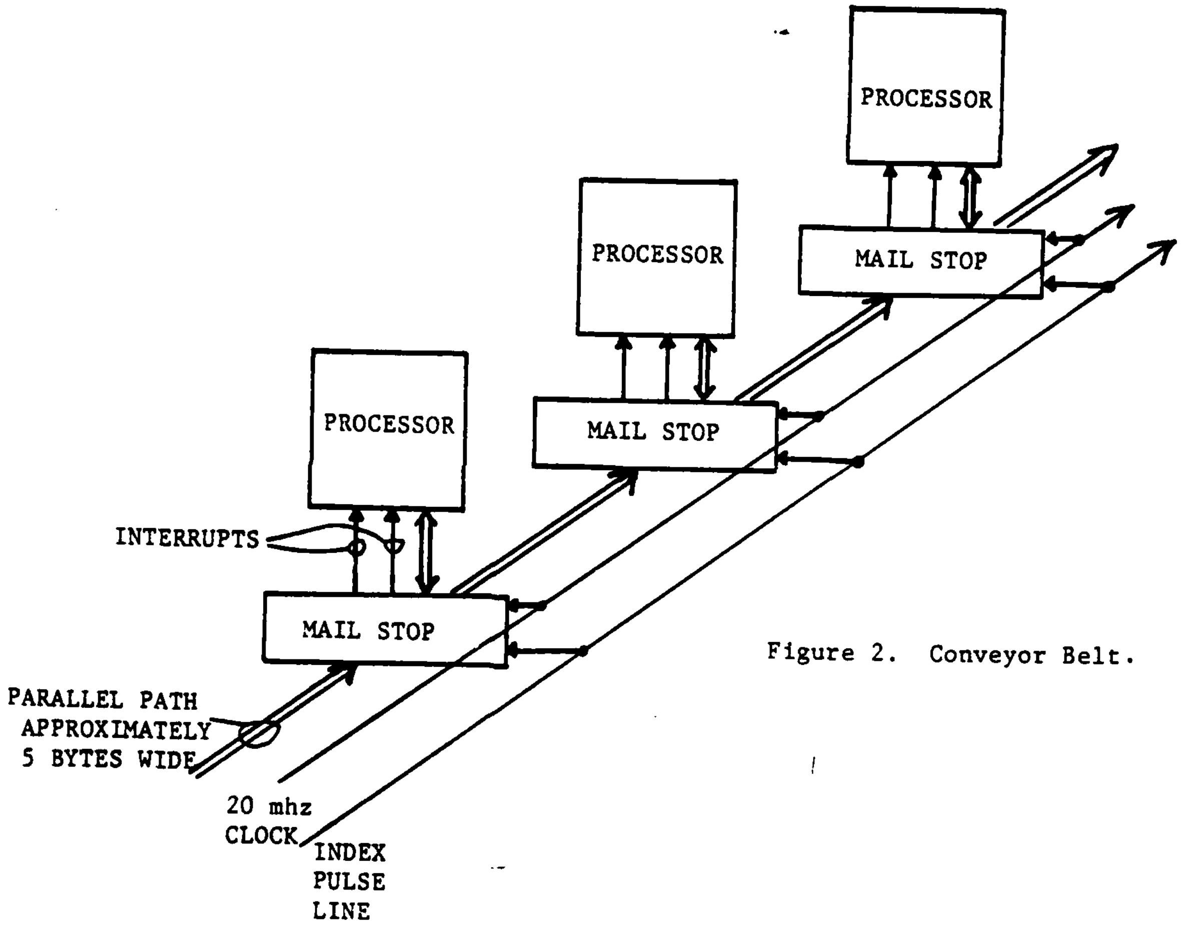

Figure 2. Conveyor Belt.

speed disk interfaces, are additional mail stops on the conveyor belt.

Each mail stop (Fig. 3) is associated with a processor, and is physically a part of that processor's PC board. Mail stops are connected together over dedicated backplane bus lines. Each mail stop is a high-speed synchronous latch capable of switching data between the processor and the conveyor belt. While the optimal width of the conveyor belt has not been determined, we are presently conducting timing studies based on a width of four fields of between 8 and 10 bits each: source ID, destination ID, data, and control.

Conceptually, the conveyor belt's role is to accept a message from a processor and deliver it to another directly or indirectly referenced processor, or population of processors. Ideally, we would like the conveyor belt to serve as a non-blocking message switcher, i.e., one in which n/2 simultaneous processor-processor conversations could be in progress at maximum Z80A rates. This would give each processor the illusion of having instant access to any other processor.

As it turns out, this ideal is achievable. Aside from DMA, the Z80A's highest memory or I/O data transfer rate is one byte per 5.25 microseconds (achieved during several of the block memory-move instructions). This is a hardware limitation of the Z80A, and cannot be improved upon by clever programming techniques. To act as a non-blocking message switcher, the conveyor belt needs only to make one complete revolution every

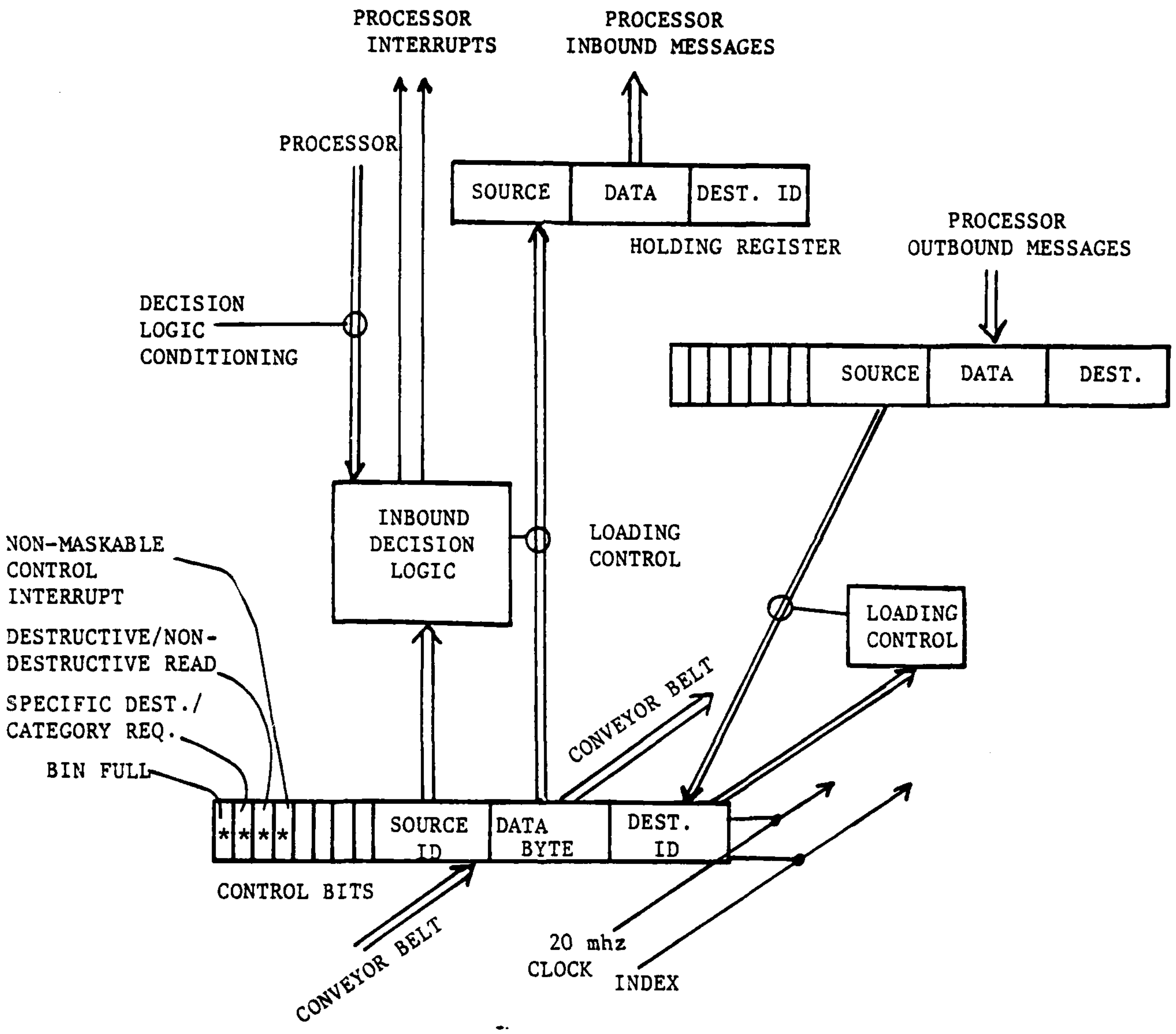

Figure 3. Mail Stop.

5.25 microseconds so that a specific position on the belt (a bin) will always be available for each processor's next memory transfer every 5.25 microseconds. Conventional high-speed digital electronics can support the approximately 50 mhz shift frequency required for such a conveyor belt. While engineering and economic considerations might dictate a somewhat slower conveyor belt in the 20 mhz range, even at a lower rate, most forms of interprocessor communication can proceed at full Z80A rates.

The conveyor belt is synchronized by two system-wide control lines, the conveyor belt shift clock, and the index pulse. The shift clock controls the basic movement of data into and out of each mail stop, and hence around the conveyor belt. The index pulse is emitted once per complete revolution of the belt, and signifies to each processor that its own bin is under the processor.

Each processor owns the bin indicated by the index pulse. When this bin is at the processor, any data waiting in the STAGING REGISTER will be taken onto the conveyor belt. The staging register, loaded byte-wise by the Z80A at its convenience, and armed when its data field has been loaded, serves to synchronize the otherwise asynchronous operations of the processor and conveyor belt. Outbound data will only be accepted when the processor's bin is flagged as empty by a bit in the control byte (i.e., if the message is not consumed by the intended destination, it will be retained on the belt, unless the originating processor has indicated that it wishes to intercept its own transmission if not consumed in one revolution).

Outbound data requires only a go, no-go decision about whether the bin is free to accept the contents of the staging register. For the inbound pathway, each mail stop requires a small amount of high-speed decision logic for intercepting conveyor belt messages directed at its processor. In addition to its numerical address on the conveyor belt, each processor can advertize a category code. When armed, this category code will accept any message whose destination field matches the code, permitting call-by-capability in addition to call-by-name.

When deemed appropriate, a conveyor belt message is lifted off the belt into the processor's HOLDING REGISTER, and the bin from which it came marked as empty. It is appropriate to lift a message into the holding register only when the holding register is empty and the inbound decision logic determines its processor to be an appropriate receiver of the message. The processor can be an appropriate receiver of a message if (1) the message is directed to the processor by direct numerical address, (2) the capability code of the message matches the processor's capability code, or (3) the processor's own message has arrived back at the processor after one complete revolution on the belt without being read. Each of these three receive conditions can be selected or deselected by the processor via processor-writable control bits in the inbound decision logic. When none are enabled, the mail stop will accept no conventional messages.

In addition to these three receive conditions, there is a fourth condition used in conjunction with high-speed block bursts between a pair of processors. In this mode, neither processor of the pair will be inspecting or setting any conveyor belt field but the data field. The receiver must therefore be in a mode which excludes all inbound messages other than those originating from the processor with which it is communicating. To support this private conversation mode, a fourth, overriding component of the inbound decision logic permits the receiver to identify an exclusive source of inbound messages. When in this mode, only a message whose source ID matches the contents of the EXCLUSIVE SOURCE register will be intercepted by the inbound decision logic.

When a conventional message is accepted into the holding register by the inbound decision logic, the BELT DATA AVAILABLE status flag is set, and a maskable interrupt generated. The processor can then inspect the message at its convenience by reading the holding register contents as a group of input ports. For block burst mode, in which both the sender and receiver are executing block instructions (and have disabled their interrupts), the inbound decision logic will also control the PROCESSOR WAIT line to provide hardware synchronization between the processor and belt.

Inbound messages can be read either destructively (i.e., consumed) or non-destructively (i.e., noted) by the mail stop, according to another control bit associated with the message. Destructive reads are used for one-one conversations, while non-destructive reads are used for broadcasting messages to the population at large.

For absolute external control, there is a class of conveyor belt control messages that will be unconditionally accepted by a mail stop. Some of these can be directed at a specific processor or class of processors, while others can be broadcast to the population at large (i.e., are not consumed by mail stops, but instead passed along). All control messages release any processor-wait condition, and generate a non-maskable processor interrupt to bring the processor back to its operating system. In this way, the controlling computer maintains ultimate control over ZMOB.

5. Patterns of Use

ZMOB will be a general purpose research tool for distributed and autonomous, parallel computating. As mentioned, each processor itself would be powerful enough to run its own operating system with text editors and high level languages, if it were a stand-alone personal computer attached to a floppy disk system. (For example, with a 48K memory, each processor would be capable of supporting a PASCAL compiler.)

In the vision relaxation applications for which the machine is to be used initially, each processor will be running the same code on its own subregion of the 512 by 512 pixel image. In a large relaxation algorithm, each pixel will be represented by, say, a 10 byte probability vector, meaning that each processor will work on 10,240 bytes of image data. This data, together with the relaxation algorithm's code and constant data, will be shipped to each processor at very high speeds over the conveyor belt during initialization. The code and constant data can be loaded at the 5.25 microsecond rate of the Z80A by having the external computer load one byte of data onto the belt in non-destructive read mode each 5.25 microsecond. All processors will note and store each byte via their high-speed block input instruction loops. This means, for example, that all ZMOB processors could accept a 10,000 byte program in just over one tenth second, assuming a conveyor belt speed of 20 mhz. After loading the program, the external computer loads the image data at whatever rate it is able. In this mode, if capable of the high data rate, the external computer could load each revolution of the conveyor belt with the next 256 bytes of image data, each directed to a different processor. Assuming the conveyor belt runs at 20 mhz, and that the external computer is capable of meeting this data rate, the 2.5 million bytes of a 512 by 512 pixel image of 10-byte-deep probability vectors can also be loaded in slightly over one-tenth second. After processing, delivery of results would occur at a comparable data rate.

The vision applications can be characterized as highly-parallel, nearly synchronous computations, not dissimilar to those for which ILLIAC IV was designed. However, these applications use ZMOB in a highly structured way. Another obvious mode of operation is one in which each processor runs its own expert code, and the machine is used more as a population of experts in the MICROPLANNER [8], CONNIVER [9], or ACTORS [10] paradigm. We expect much interesting research to open up in this area.

Another mode of operation would segment ZMOB into fiefdoms. Each fiefdom would be a cluster of processors, governed by one agreed-upon distinguished member. This member would be responsible for one ongoing computation, and would use his serfs primarily as extended memory which he could page in as needed. Since the conveyor belt has been designed to be responsive to high-speed data bursts among processors, we will be able to develop a very fast paging system capable of paging rates equal to or surpassing good disks (i.e., no seek latency, but somewhat slower data rates). In this pattern of use, for example, we might run LISP on ZMOB by creating a fiefdom for the evaluator, one for the garbage collector, one for the scanner, one for a real-time debugger/monitor, and so on.

In a more conventional pattern of use, only one of ZMOB's processors would be distinguished as the central computing processor, and all other processors would support high speed paging and memory management for that one processor. This would make all 8 or 12 megabytes of high speed memory directly accessible, and would resemble a large conventional computer with very fast paging potentials. However, since a single Z80A is admittedly not a high throughput machine, it will probably turn out that ZMOB will seldom be used in this mode.

6. Conclusion

It is anticipated that ZMOB development will be in full swing by late spring 1980. Before that time, we hope to have a prototype system of 4 to 8 processors running on a small conveyor belt. The project will be supported by several faculty and graduate students, and will hopefully be in relatively complete form by spring 1981. During development, extensive software development tools (assemblers, higher level system languages, simulators, debuggers) will emerge. Hopefully, the machine will also stimulate and make possible new theories of distributed problem solving and parallel computing.

Acknowledgements

Azriel Rosenfeld supplied the vision application which stimulated initial thinking about ZMOB. Through discussions, John Bane, Randy Trigg and Milt Grinberg have greatly contributed to the ideas described here.

7. References

1. Duff, M. J. B., and Watson, D. M., The Cellular Logic Array Processor, Computer J. 20, 1977, 61-72

2. Slotnick, D. L., Research in the Application and Design

Refinement of the Massively Parallel Processing Computer (MPP), Quarterly Progress Reports, University of Illinois, Urbana, IL, Oct. 1977 ff.

3. STARAN, in Enslow, P. H., Jr., ed., Multiprocessors and Parallel Processing, Wiley, NY, 1974

4. Fuller, S. H., et al., A Collection of Papers on CM*: A Multi-Microprocessor Computer System, Carnegie-Mellon Computer Science Memo, 1977

5. Dennis, J. B., Misunas, D. P., and Leung, C. K., A Highly Parallel Processor Using a Data Flow Machine Language, CSG Memo 134, Lab. for CS, MIT, 1977

6. Rosenfeld, A., Iterative Methods in Image Analysis, Pattern Recog. 10, 1978, 181-187

7. Prime Computer, Inc., Prime Net Reference Manuals

8. Sussman, G., Winograd, T., and Charniak, E., Microplanner Reference Manual, MIT AI Memo 203a, 1971

9. McDermott, D., and Sussman, G., The Conniver Reference Manual, MIT AI Memo 259a, 1972

10. Hewitt, C., Viewing Control Structures as Patterns of Passing Messages, MIT AI Memo 410, 1976