ZMOB: A NEW COMPUTING ENGINE FOR AI*

Chuck Rieger, Handy Trigg, Bob Bane

Maryland Artificial Intelligence Group
Department of Computer Science
University of Maryland
College Park, Maryland 20742

ABSTRACT

A new research multiprocessor named ZMOB is described, and its significance to AI research is surveyed and characterized. ZMOB, under current construction and scheduled for completion late Fall 1981. is a 256 processor machine with a high speed ,interprocessor communications system called the "conveyor belt", a non-blocking, 20 megabyte/second message switcher. Because of its large number of processors, high computational throughput (100 million instructions/sec), large cumulative high speed memory (16 megabytes), high interprocessor communications bandwidth, and external communications and sensing channels, ZMOB opens some new areas of exploration in object-oriented modeling, knowledge-based and domain expert systems, intelligent sensing and robotics, and distributed perceptual and cognitive modeling. The paper attempts to blend a description of ZMOB's hardware with ideas about where and how it fits into various types of AI research.

1. Introduction

AI is not in need of new computing hardware to solve its basic questions. However, new hardware cannot be dismissed as being irrelevant to progress in AI, because:

1. Beliefs about the nature of computing hardware available influence a researcher's ability to conceptualize data and process models of intelligence. While a researcher can certainly imagine methods of modeling which require unusual hardware, he may never discover the interesting issues because of his inability to see beyond the first round of ideas. Like It or not, AI is an experimental science which relies heavily on feedback from the implementation level to the conceptual model level.

2. Certain AI methods requiring heavy symbolic computation have not been adequately explored simply because the architecture or computational capacity of the supporting computers is not adequate. Examples are large pattern-directed rule systems, and certain low and high level vision techniques, among many others.

3. Certain well researched AI methods. recognized as fruitful conceptual approaches to various classes of problems, are inherently not designed for conventional machines, and would become practical, real-world systems if implemented on hardware better suited to tneir needs. Distributed and "expert" systems which rely on object-oriented programming styles are generally good examples of this category.

* The research described here is funded by the Air Force Office of Scientific Research under Contract AF05R-80-0270. As of March 1981, all ZMOB design and debugging was completed, and a two-processor prototype was running. The full machine, budgeted at approximately $120K hardware cost, is expected to be running by late Fall 1981.

This paper is about a new research computer designed to help stimulate new ways of thinking about AI modeling, and to help advance certain existing computing methods by removing some of the computational barriers that limit these methods usefulness. We offer this research not as a source, but rather a gateway to some new ideas about AI model building. In writing a paper about hardware's role in higher level model building, we run the risk of overselling the hardware per se. without demonstrating its usefulness to model building and theory development. To minimize this risk, we have,attempted to blend a description of the machine's architecture, including enough detail to give the reader a complete picture of the machine, with brief tours through the various realms of AI model building which may profit from this architecture. Our hope is to snow that new machine architectures, if developed in a way that is sensitive to current modeling directions in the field, can be an important link in the larger basic research feedback loop that gradually converges on good AI theories.

2• ZMOB and Distributed AI

ZMOB is a 256 processor multiprocessor which taxonomically falls into the same general category as CK* [13], but which also has appearances of high speed ring networks. Each of its 256 processors is a completely autonomous computer with enough memory to do significant independent high level computation asynchronously from other processors. However, each processor is intimately linked to all other processors by an interconnection system of such high speed relative to the individual processor that all 255 of a processor s companion processors appear to be equidistant from it, and, in fact, accessible in what amounts to unit time from its point of view. ZMOB is thus a mob of 256 reasonably powerful processors organized in a manner tnat superimposes no actual, or even preferred, geometry on their logical interconnectivity. As such, it is close to the ideal for systems of experts whose intercommunication geometry is governed by domain problem solvers rather than by a priori donain structure.

ZMOB differs from CM* in three significant ways. First, it possesses a larger number of processors, each roughly equivalent in power to a CK* processor. (To our knowledge, its 256 processors comprise the largest research multiprocessor yet designed and built.) Second, all memory is uniformly distributed equally among the processors (there is no high speed shared memory), giving ZMOB its network flavor. Third, the interprocessor communications strategy is completely non-blocking, with no possibility for hardware-level contention in message passing. ZMOB will support 128 pairwise processor conversations, and broadcasts from one processor to all others or a subset of others equally well.

In the next section we describe ZMOB's hardware architecture and operating systems, which will include C, LISP, PASCAL, PROLOC, and specialized vision systems. Then we survey ZMOB s

conceptual and pragmatic usefulness to several areas of AI.

3. Hang on: Here's the Hardware!

ZMOD is a large, experimental research multiprocessor under construction by the Artificial Intelligence Group within the Computer Science Department at the University of Maryland. The machine's architecture is 256 autonomous microprocessors, each with 64K bytes of 300 ns memory and capable of approximately 400,000 instructions per second, connected together via a high speed communications ring called "the conveyor belt". The conveyor belt, a 48-bit wide,,10 mhz shift register, comprises 257 "mail stops". Each 48-bit-wide mail stop is associated with a ZMOB processor (except for the 257th, which is associated with ZMOB's external interface), and represents one stage in the conveyor belt. Because or the conveyor belt s high speed relative to the needs of any individual processor, and because of each mail stop's high speed message pattern matchers, the conveyor belt is a nearly perfect interprocessor communications medium. In particular, the conveyor belt gives prompt enough service for every processor to be given "unit time" message delivery to another arbitrary processor or Bet of processors.

In addition to its conveyor belt interface, each processor has a high speed parallel and a serial interface to the outside world for applications involving remote sensing, control, or communications. A special 257th processor interfaces the ZMOB conveyor belt to a PDP Unibus.

At large, ZMOB executes approximately 100 million instructions per second on a 256-way distributed 16 million byte high speed memory. Its conveyor belt switches interprocessor and external messages at a rate of 20 million bytes per second.

3.1_* Mail Stops and Bins

Physically, each Mail stop is a printed circuit board which occupies one slot of the conveyor belt backplane, a 48-Dit wide circular bus which passes through all mail stops. Incoming conveyor belt signals (those from a mail stop s upstream neighbor) arrive via the mail stop s 48 inbound lines, and outgoing conveyor belt signals (those being sent to the mail stop s downstream neighbor) leave via the mail stop's 48 outbound lines. One 40-bit "bin" (i.e.. one message) passes through the mail stop each 100 ns, as directed by the 10 mhz synchronous master system clock which is distributed to each Mail stop over precisely matched cables.

Each ZMOB processor occupies a second printed circuit board which is physically adjacent to a mail stop board. Each mail stop communicates with its associated ZMOB processor via a 20 line interface that allows the processor to view its mail stop as a collection of 8-bit registers which, when read from and written to, cause mail and status information to flow to and from the conveyor belt.

Logically, the conveyor belt is thought of as a collection of bins wnich move around the loop through all mailstops each revolution. Bins carry messages, and each processor permanently owns one bin. Once, per conveyor belt revolution, every processors bin simultaneously arrives back at the processor s mail stop. This event is signaled to all mail stops by a master "index pulse", generated by the master clock circuitry and distributed to each mail stop as with tne master shift clock.

The mail stops are analogous to the ola freight train mail hooks. When the processor wishes to send a message, it gives it to its mail stop, which holds it until the proper moment for injection on the conveyor belt (i.e.. index pulse time). When the conveyor belt bin permanently owned by a processor arrives back at that processor's mail stop empty, and if the mail stop has been given a message to send, then the message will be injected onto the conveyor belt in the processor's bin. During all other moments (i.e., when processors' bins are in transit around the conveyor belt), each mail stop performs high speed pattern matching to detect arriving messages intended for its processor. The mail stop can extract one message from the conveyor belt and hold it until its processor has taken it.

Each mail stop (hence processor) has a unique conveyor belt address (0-256), which can be used to identify the intended receiver(s) of a message. In addition, each mail stop can be instructed by its processor to watch for a specific pattern of 1's, O's, and "don't care's" in messages' destinations fields, and to receive any messages matching that pattern. This is a basic mechanism upon which more elaborate pattern-directed invocation techniques and hierarchical processor organizations can be realized.

3.2.. Messages

When a message is sent, it will always contain the address of the sender, the data, and information describing the intended receiver(s). There are four modes for describing the intended receiver(s):

SEND TO PROCESSOR BY ADDRESS

SEND TO PROCESSOR BY PATTERN

SEND TO ALL PROCESSORS

SEND TO SET OF PROCESSORS BY PATTERN

Each message is sent under exactly one of these four modes. In the first two cases, where there is only one intended receiver, the receiving mail stop will consume the message. In the la3t two case3, a receiving mail stop will make a copy of the message, but allow it to continue for all to see. In this latter case, it is incumbent on the sending processor to consume its own message after one complete conveyor belt revolution. Doing 30 is called "readback".

Each mail 3top is capable of buffering one message in the outbound direction (processor to conveyor belt), and one message in the inbound direction (conveyor belt to processor). Messages will be received by a mail stop only if its pattern matchers are enabled and deem the message appropriate, and its inbound buffer is empty. A received message will generate a processor interrupt, then remain in the mail stop's inbound buffer until explicitly read by the mail stop s processor. Messages on tne conveyor belt intended for a processor whose mail stop's inbound buffer is full will simply continue circulating on the conveyor belt until the mail stop finally reads and consumes them, or until their senders consume and remove them. When the mail stop's outbound buffer is emptied as the result of injecting the message it contains onto the conveyor belt, a second type of processor interrupt is generated. The mail stop is therefore a full-duplex protocol,, where separate message streams can be in progress both inbound and outbound.

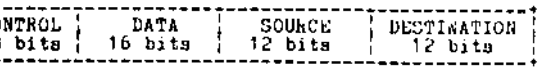
A mail stop's receiving pattern matchers can be further conditioned to accept only messages from a specific source processor, identified by its conveyor belt address. This "exclusive source mode allows for the uninterrupted transmission of data streams between two processors, or between one source and a group of receivers.

In order for absolute control commands from the conveyor belt a Unibus interface to have guaranteed delivery to any or all processors, each mail stop has a separate logic section that is capable of detecting special control" messages from the mail stop associated with the Unibus interface processor, the only mail atop capable of sending control messages. Control messages are subject to the same inbound pattern matchers as normal messages, except that (1) they cannot be selectively excluded the way normal messages can, and (2) they do not interact with the mail stop's inbound buffer. This transparency allows absolute access to any or all processors from the external

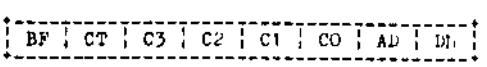
conveyor belt interface, while not interfering with the flow of normal conveyor belt messages.

3.3 Conveyor belt Format

The 48 bits of a conveyor belt message are appropriated into four fields:



The CONTROL bits are defined as follows:



BF - Bin Full
CT - Control Message
C3-CO - Control Code
AD - All Destinations
DM - Destination Mode

When sending a message, the processor has freedon in setting the AD and DM bits to any values. The AD and DM bits describe the mode under which the message is being transmitted, and are inspected by the inbound pattern matchers of all potential receiving mail stops:

AD	DM	INTERPRETATION
0	0	SEND TO PROCESSOR BY ADDRESS The message is intended only for the processor whose address is in the destination field.
0	1	SEND TO PROCESSOR BY PATTERN The message is intended for the first processor whose posted Pattern matches the destination field.
1	0	SEND TO ALL PROCESSORS The message is intended for all processors. (The destination field is to be ignored.)
1	1	SEND TO SET OF PROCESSORS BY PATTERN The message is intended for all processors whose posted pattern matches the destination field.

The BF and CT bits are not controllable by the processor. The BF flag is generated entirely by mail stop hardware logic, and the CT bit can only be generated by the 257th mail stop.

The 16-bit DATA field provides for single or double byte messages. If CT is set then bits C3-CO are interpreted as a control code. Otherwise they can be used as extra data bits. The SOURCE and DESTINATION fields have been allocated 12 bits for possible future increases in the number of processors.

3.4. Processor Interface

The processor views its mail stop as a group of 16 8-bit memory-mapped I/O ports, each of which accesses part or all of one of the mail stop s registers. All reading and conditioning of the mail stop's current state, and all data movement between the mail stop and processor occurs by reading and writing to these registers via the 16 8-bit ports. The mail stop can also generate four types of interupt to signal the processor that it requires service: INBOUND BUFFER FULL, OUTBOUND BUFFER EMPTY, READBACK, and CONTROL.

3.5.* ZMOB Processors

Each ZMOB processor is a Z80 microprocessor with the following support hardware:

1. 63K bytes of high speed compute memory
2. 1K bytes of resident bootstrap operating system read-only memory
3. Interface logic to conveyor belt
4. A high speed hardware integer multiplier
5. A 32 bit floating point processor chip
6. A high speed 8-bit parallel and a serial external interface.

When ZMOB is first powered on, all processors go through identical reset sequences which initialize all processor and associated mail stop logic. Each processor then enters its resident bootstrap kernel, which awaits commands from the Unibus interface. Support software on the VAX then sends operating software to each processor individually, or to all at once, in case all will be running identical operating systems.

4. ZMOb Software: high Level Strategies

4.1. Conceptual Organizations

The particular operating system regime under which ZMOB processors are coordinated depends largely on the domain. We 3ee five conceptually different control regimes for AI modeling:

1. ACTUAL SYNCHROMY. In this most rudimentary system, each processor performs fixed functions in precise (e.g., conveyor belt revolution time precision) lockstep with all others, but on different data points. This is an architecturally uninteresting use of the machine, but is applicable in such areas as cellular automata modeling and neural simulation, certain numerical methods in perceptual modeling, and so forth. In these applications, either high efficiency, or high precision in simulation timing requires precise synchronization.

2. LOGICAL SYNCHRONY. Each processor runs an identical operating system capable of performing a fixed set of functions in logical (but not necessarily actual) synchrony with all others, eacn working on a piece of a large data space distributed among processors in advance. Examples of this mode are models of low level vision or speech (where edge detectors or acoustic processors all proceed in parallel on different data), and in parallel databases storage and retrieval, where all processors carry out (possibly intelligent) hashing and pattern Matching on different segments of the database (see [9] for example).

.5. UNIFORM MULTIPROCESSING. Each processor runs an identical operating system, but one capable of general computing or problem solving. Each processor is statically or dynamically assigned a subtask, which it then solves asynchronously from all others. A large number of AI methods falls into this category, since each processor is, in effect, a general problem solving agent.

4. ARTICULATED MULTIPROCESSING-. Distinct domain experts reside in various processors, or sets of processors. Each expert is capable of autonomous solution of problems in its domain, which may have been sent to it via a pattern-directed invocation scheme. Systems of experts such as those encouraged by MicroPlanner and Smalltalk fall into this category.

5. HYBRID SYSTEMS.. The machine is logically segmented to support any or all of the above modes simultaneously, because of the nature of the conveyor belt, one-quarter of the machine could be devoted to processes requiring actual synchrony, one-fourth to those running logical synchrony, one-fourth to uniform multiprocessing, and one-fourth to articulated multiprocessing. Illustrations of hybrid use can be taken from nearly every area

of AI, including robotics, expert systems, speech systems, natural language systems, and game playing systems. In a typical segmentation, some processors would handle sensory and perceptual information, others the logical database, others the domain problem solving, and others the effector or response control.

4.2. Four Processor-Resident Operating Systems

In anticipation of such variety of modeling needs, we are developing four higher level processor-resident environments, all of which rely on a single, lower level kernel system. At poweron, the kernel system will be broadcast from the VAX to all processors. Then, at user request, various processors will receive one of the four higher level environments:

1. c. In this system, the processor supports a C runtime environment, and runs C code which has been developed and cross-compiled on the VAX.

2. LISP. The processor runs a full Z80 LISP system [3], and communicates via S-expressions passed as ASCII strings among processors. The issue of pointer passing on a completely distributed CONS node apace will have to be addressed before other forms of interprocessor LISP communication can be achieved.

3. PASCAL. The processor supports a PASCX runtime environment and either runs cross-conpiled PASCAL code, or interprets P-codes.

4. D+D. The processor ,,runs a dedicated database and demons" operating system. This special-purpose system allows the processor to participate as a piece of a larger pattern-directed, intelligent database, which stores information in LISP S-exnression form (although encoded differently for efficiency. The D+D system will store both constant S-expressions (facts) and schematic S-expressions (demons), and will respond to fetch, store, and trigger commands sent to it via the conveyor belt.

The C, LISP and PASCAL systems already exist for ZMOB, since each processor is capable of running CP/M (a nearly universal machine-independent operating system for the Z80). D+D is under current development in Z80 assembler, using the VAX Z8O assembler and simulator, and the VAX process-oriented conveyor belt simulator specially written for ZMOB development.

The resident low level operating system kernel common to all higher level environments is in effect an event-driven task management system which carries out all low level interprocessor handshaking and data transfers. The interrupt-driven task queues at this low level support a succeso/failure/contingency communication system similar in some respects to SAIL's process management. We have not yet completed, this system, and are looking at Rashid s IPC [19] as an eventual augmentation of the present kernel.

We turn now to some specific AI modeling where ZMOB will have impact. We have organized this discussion around AI areas, but point out specific systems where appropriate.

5. Where ZMOD Fits: Areas and Issues

5.1 Object-Oriented Programming and Experts

The concept of object-oriented programming dates at least from the work of Alan Kay [14] and Carl Hewitt [11] and more recently is the dominant systems organization paradigm of the Smalltalk system[12]. In this style of programming, active knowledge is organized around cooperating experts CALLED objects. 'The hierarchical organization of objects into classes, subclasses and instances both aids the user in creating new objects and facilitates elegant communication among objects in the form of message passing.

ZMOB's distributed, geometry-independent nature makes it an ideal parallel machine for implementing systems organized in such a manner. The localized processing required for the object-oriented approach is encouraged by ZMOB's distributed memory. At the same time, its ability to dynamically link arbitrary pairs of processors in direct communication allows for sophisticated message passing capabilities.

We are currently in the process of designing and implementing a system for computer aided design,and simulation of mechanisms (CADSOM) on ZMOB [29]. This system possesses many of the features outlined above including separate processors as experts for different parts of a user defined mechanism and message passing as a means for real-time graphical simulation of the kinematics of the mechanism.

Briefly, the operation of our CADSOM system is as follows. The user, say a mechanical engineer, begins with a rough design of a mechanism in mind. He or she then interactively identifies parts from a frame-like database of primitive part descriptions. These can then be further specified as to size, weight, shape, etc. while viewing the object on a graphics terminal. As a part is described it is assigned to a processor and as the links between parts are described by the user, communication paths are established among the processors. Finally, when the user has described the complete assembly, he or she may specify an initial velocity or force upon some part at which point the system performs a graphical simulation in real-time of the kinematics of the mechanism. It is during the simulation phase that truly parallel object-oriented processing occurs with part experts interacting both to compute velocity, acceleration, and force vectors for the next incremental time step and to detect obstructions to their motion.

5.2. Parallel Pattern Matching and Production Rule Systems

Though the production rule system has been popular and successful as a design methodology especially for large-scale .-knowledge , engineering tasks for some time (see [27], [7], [31]) it has only been recently that the problem of implementing such systems in parallel has been confronted. In particular, Forgy in [9] has designed a "production rule system based In part on ideas from data flow theory. Though to our knowledge, hi3 system has not yet been implemented on a parallel machine, many of his algorithms including those for pattern matching are inherently parallel.

A fertile area of study for ZMOB is this topic of parallel pattern matching specifically addressing ways of avoiding the need for a large global database of patterns. Distributing the large database over many processors raises additional problems. *One* in particular is that of bottlenecking at processors who have a disproportionate share of requests on their portion of the database. Thus the design of algorithms for efficiently and dynamically reconfiguring the database organization is an important area of research on ZMOB.

5.3. Cognitive Modeling

Many of the past and present theories advanced as plausible cognitive models of natural language understanding, causal reasoning, inference, and problem solving directly or indirectly posit distribution of functions in ways appropriate to ZMOB. Parallel matching of KRL frames [5], scripts in FRUMP and related systems [26] are examples where theoretically Quantized knowledge unit3 could be experimentally distributed across processors. The actors in Meehan's story generator, and entities in complex social or political sinulation models are examples where independent problem solving agents could be run by separate processors. Individual, word experts in the Word Expert Parser *{[22]* and [28]) are a perfect fit

for ZMOB. In that theory, each word is modeled by a self-contained word expert, LISP code that interacts with other experts during the comprehension of a sentence in context. Our CSA representation ([21] and [20]) and its simulation system directly segment the world of causality into events and event clusters which could be parallelized directly into autonomous agents. In fact, the whole area of causal reasoning in the physical world seems directly suited to distributed modeling. In all these cases, ZMOB could be playing the dual role of helping shape theories about distribution of cognitive function, and of making experimentation more reasonable. Experience with the *WEP* and CSA systems, for example, is that interesting research has sometimes not been attempted because of our beliefs and frustrations about computational limits of the conventional machines we are using.

5.4. Robotics and Real Time Sensing Systems

ZMOB's real-time performance and external serial and parallel channels per processor make it ideal for applications involving coordinated and articulated control of effectors and/or synthesis pf 3ensory information. As the NBS robotics lab [2] and others have found, multiple fixed-function processors can bring sophisticated vision, sensing, and joint control to nearly any level of desired motor performance. In an automated shop, for example, where a number of tools and their associated vision and manipulator systems must run in coordination, ZMOB processors could be assigned in modular clusters that communicate both at the intra- and inter-group levels. In one module, for example, several processors would perform region-wise feature extraction in parallel, feeding results to a separate scene-constructing processor, which then directed the team of manipulator trajectory computing and joint controlling processors. Another processor in the cluster could scan for certain error or contingency conditions, and a final processor could coordinate its group's efforts with other group coordinators. In a large job shop, human operator stations could be tied in by other processors.

In sensing environments, ZMOB could assign processors to fixed sensor clusters in a complex environment, such as Three Mile Island. Each processor would contain both physical and causal models of the segment of the environment it sensed, and the relation of that environment to the site at large. Given powerful enough causal models and the current operating context and user intent (e.g., during dynamic maneuvers such as peak load power up), individual processors could intelligently coordinate their environment with the larger environment, report abnormalities with annotations about probable causes, and even take local corrective procedures, then inform superior processors of the changes. Distributed intelligence in a complex sensor/effector system like TMI is an attractive solution to real-time critical decision making, and ZMOB seems ideally suited for such distribution. A central research topic here is to develop the common language of causality through which processors communicate. With its effective interprocessor communications bandwidth of 160 megabaud, ZMOB's potentials for distributed real-time decision making are considerably higher than even the fastest traditional networks. Also, because of the large number of processors, redundancy either at the decision making or sensory level is feasible.

5.5. Management of Large Information Spaces

In [24] we have described the concept of an I-SPACE, and an I-SPACE project in progress at Maryland. An I-SPACE is an information hub for large societal or institutional organizations where there is a large quantity of information and/or a large number of tools for manipulating that information, as well as a need to interface a wide variety of users to both the information and tools in real time. The Goddard I-SPACE is perhaps the archetype of all I-SPACEa, in that the user group is quite diverse (administrators, scientists, engineers, technicians, schedulers, etc.), and in that the information content of the system at large is both tremendous and highly distributed (science and engineering databases, real-time data acquisition systems, real-time control systems, planning and administrative databases, and so forth). The keys to the I-SPACE concept are uniformity in information presentation on high resolution TV screens, and flexibility in synthesizing information in user-idiosyncratic ways from all parts of the I-SPACE for simultaneous display on his screen.

While the user interface in the Goddard I-SPACE system is reminiscent of the PIE system. [10] the real-time requirements of accessing ana synthesizing information from remote hosts, administrators' desktop computers, real-time satellite links, and local planning databases call for much more computational power than PIE, or, indeed, than most computing facilities possess. Because of this, ZMOB will play a prominent role, called the I-BANK, in the Goddard I-SPACE. In this role, ZMOB's processors will run dedicated, intelligent links to the outside world (remote hosts, telephone links, satellite links, etc.) that comprise the I-SPACE. The system s VAX-based user interfaces, running frame-like packets which put out real-time control and information requests into the I-SPACE, make demands on ZMOB as a gateway to the outside, thus distributing most or the system s time-critical scheduling to ZMOB *a* processors. We see this type of architecture as mandatory for large-scale information systems.

5 .6. Distributed Databases and Formal AI Research

Preliminary design of a predicate logic system to be implemented on ZMOB is currently in progress at the University of Maryland. Based on such foundational worn as [15], [6], and the set-oriented extensions of [18], this system features a distributed relational database and the predicate logic language PROLOG [30] used as an interface to the database for querying and other purposes.

Initially, three types of processes are envisioned: the search space process which accepts the user s database query, divides it into problems and starts up several problem solving processes to handle them; the problem solving processes which run PROLOG and build tasks for the database processes; and the database processes which (initially) handle one relation of the entire database apiece and perform lookup, updating, and unification. In the future, the ability to perform efficient reorganization or the database will be added in order to spread an overused relatiop among several processors and to group several relations in the same processor. ZMOB s architecture allows complete freedom to effect such a reorganization while its message passing protocols allow problem solving processes to communicate by capability with data base processes, a necessity for efficient database lookup.

5.7. Heuristic Search

The problem of searching a large state space has long begged for a parallel solution. Several algorithms for doing this have appeared in the literature ([1] [4] , [8]) but most have not yet been implemented on true parallel hardware. Those that have been implemented are not as useful as they could be. For example. Baudet s parallel version of alpha-beta exhibite a better than linear speed-up with two or three processors, but improves very little with more than five. ZMOB s flexibility makes it easy to implement these algorithms, compare their performance, and develop new methods from that experience.

5.8.. Computer Vision

Some preliminary work has already been done in the area of image processing on ZMOB. In [23], a vision system running on ZMOB is described which divides tne 512 by 512 pixel image into 32 by 32 regions, one per processor. Timing simulations showed that a simple 3 by 3 averaging of the entire image could be performed in about 2 fifths of a second while elaborate relaxation algorithms

involving 10 labels per point could complete an iteration in about 100 seconds. Such timing estimates show that orders of magnitude speedup can be achieved using ZMOB in the vision domain. [16] provides further timing estimates for picture processing algorithms including image loading and unloading times, and notes that for most non-trivial algorithms ZMOB effects as much as a ten-fold speedup over execution on the host, computer (in this case, a VAX 11/780). In [17] these results are compared with those of possible implementations on other parallel architectures, in particular, switching networks.

5.9. Other Areas

In *[25]* the appropriateness of ZMOB for research in cellular automata is discussed, specifically addressing the problems of reconfigurable parallel computers. Other future application areas include numerical analysis, network topology and performance evaluation, and neural modeling. Finally, ZMOB could also function as a microprocessor instructional machine for students or as a message switching station for large computer networks.

6. Summary

ZMOB is scheduled to become fully operational in late Fall 1981. As we emphasized in the introduction, it will directly solve no theoretical AI problems. But it will, we hope, encourage deeper research into existing areas where computational limitations have traditionally posed serious barriers, and stimulate new and more ambitious ways of thinking about AI and cognitive processes. It will, we hope, represent a new, important link in the design/implement/test feedback loop that is so important to AI.

REFERENCES

[1] Akl, S. C, D. T. Barnard, and R. J. Doran, Design, Analysis, and implementation of a Parallel Alpha-Beta Algorithm Queen's University, Technical Report 80-98, 1980.

[2] Albus, J. S., A. J. Barbera, M. L. Fitzgerald, R. N. Nagel, G. J. Vanderbrug, and T. E. Wheatley, A Measurement and Control Model for Adaptive Robots, in Proceedings of the 10th International Symposium"—cm—ITOu's^ Trail Robots, 1980.

[3] Allen. John, The TLC LISP User's Manual, The Lisp Company ,1980

[4] Baudet. C. M., The Design and Analysis of Algorithms for Asynchronous Microprocessors, Carnegie-Mellon University CMU-CS-78-116, 1978.

[5] Bobrow, D.G. and T. Winograd. An Overview of KRL, a Knowledge Representation Language, Cognitive Science 1, (t), 1977.

[6] Chang, C. L. and R. C. T. Lee, Symbolic Logic and Mechanical Theorem Proving; academic Press, New York, 1979

[7] Duda,, Hart, and Nilsson, et al, Development of a Computer-Based Consultant for Mineral Exploration. Annual Report, Stanford Research Institute, 1977.

[8] Fishburn, J. P., R. A. Finkel, and S. A. Lawless, Parallel Alpha-Beta Search on Arachne, pp. 235-243 in Proceedings of the 1980 International Conference" on" Parallel Processing, 1980,

[9] Forgy. C. L., A Production System Monitor for Parallel Computers, CMU 161, April, 1977.

[10] Goldstein, I.P. and D.G. Bobrow, Descriptions for a Programming Environment, in Proceedings of the First Annual National Conference on Artidicial Intellengences stanford ca Aug

[11] Hewitt, C, Viewing Control Structures as Patterns of Passing Messages. Artificial Intelligence 8, pp. 323-364, 1977.

[12] Ingalls, D.H.H., The Smalltalk-76 Programming System Design and Implementation, in Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, 1978

[13] Jone3, A. K. and P. Schwarz, Experience Using Multiprocessor Systems - A Status Report, Computing Surveys 12, 2, June 1980.

[14J Kay, A.. The Reactive Engine, Ph.D. thesis, University of Utah, 1969-

[15J Kowalski. R. A., Logic for Problem Solving, North-Holland, 1979?

[16] Kushner. Todd, Angela Y. Wu, and Azriel Rosenfeld, Image Processing on ZMOD, University of Maryland, TR-987, Dec. 1980.

[I7] Kushner, Todd, Parallel Image Processing Using Switching Networks for Interprocessor Communication, University of Maryland, TR (forthcoming), March 1981.

[18] Minker, J., A Set-Oriented Predicate Logic Programming Language, TR-922, University of Maryland, 1980.

[19J Rashid, R.F., An Inter-Process Communication Facility for UNIX, CMU-CS-80-124, Carnegie-Mellon University, Jun. 1980.

[20] Rieger, C., The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief and Contextual Language Comprehension, University of Maryland TR-373, Kay 1975-

[21] Rieger, C. and M. Grinberg, A system for Cause-Effeet Representation and Simulation in Computer-Aided Design, in Proceedings of the IFIP Working Conference on Artificial Intelligence ana Pattern Recognition in Computer-Aided Design, 1978.

[22] Rieger, C. and S. Small, Word Expert Parsing, pp. 723-728 in IJCAI-79, 1979.

[23] Rieger, C, ZMOB: A Mob of 256 Cooperative Z80X-Based Microcomputers, in Proceedings of the ARPA Vision Workshop, Los Angeles, CA 1979

[24] Rieger, C., R. Wood, and E. Allen, Large Human-Machine Information Spaces, Univ. of Maryland CS TR-1024, Mar. 1981. (also submitted to IJCAI-81)

[25] Rosenfeld, Azriel and Angela Y. Wu, Reconfigurable Cellular Computers, University of Maryland, TR-963, Oct. 1980.

[26] Schank, R. and C. Riesbeck, Inside Computer Understanding, L. Erlbaum, 1981.

[27] Shortliffe, E., Computer-Based Medical Consultations - MYCIN, American Elsevier, 197b.

[28] Small, S., Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding, University of Maryland TR-954, Sept. 1980.

[29J Trigg, Randall H., A Parallel Approach to the Interactive Design and Simulation of Mechanisms, University of Maryland, TR-992, Doc. 1980.

[30] Warren, D. H. D., Implementing PROLOG: Compiling Predicate Logic Program, University of Edinburgh, Research Reports 39 & 40, 1979-

[31] Waterman, D. and F. Hayes-Roth, An Overview of Pattern-Directed Inference Systems, in Pattern-Directed Inference" Systems, ed. Waterman a Hayes-Roth,Academic Press ATadefflle Press, New York, 1978.