ANNALES
UNIVERSITATIS MARIAE CURIE-SKŁODOWSKA
LUBLIN — POLONIA
VOL. XLIII/XLIV, 12 SECTIO AAA 1988/1989

Instytut Fizyki Doświadczalnej
Uniwersytet Warszawski

M.W. KALINOWSKI

Computer Architecture, Algorithms, Computer Physics and All That

Contents

Introduction

Without going into a thorough historical review of the history of the digital computer (the other major branch of computers are those of analog design which will have to be left for another occasion), the general lineage of today's computers is traced from Babbage's predecessor mechanical machine (1834) to the first generation vacuum tube computers (MANIAC and Eniac. circ. 1940) and on to the second-generation transistor computer (Univac. IBM 1401, 2000 series, circa 1950), and third-generation integrated circuit machines (IBM 360 series. 1960), to the present LSI (large scale integration and micro-processor designs). All of these computers to date have been largely built around the Von Neuman architecture, wich will be described momentarily. Present attempts, and there are many, to develop the fifth generation of computers generally revolve around VLSI (Very Large-Scale Intergration, also to be explained shortly), and multi-processor architectures.

The advances in computer hardware design have been the result of increased speeds of operation and reduced costs of memory. The first major advance in speed, of course, came with the move from mechanical devices like Babbage's to the electronic device of the vacuum tube. The advent of the transistor meant that components could be packed much closer together. Each halving of distance between components meant a doubling of speed with which signals could be processed between them. Transistors required much lesss energy to operate and, consequently, produced much less heat than vacuum tubes. Partly because of this, they proved to be much more reliable, and their increased MTBF (Mean Time Between Failures) permitted the construction of larger, more powerful systems.

The same principles and results applied in the next stage of development as Integrated Circuits appeared on the scene. Onece again, circuitry was condensed so signals had less distance to travel, and there was less energy required, which meant fewer heat problems. The I.C. (integrated circuit) or chip at first combined within itself a half dozen of transistors or the equivalent logic of as many vacum tubes. Improved manufacturing and design techniques over the next couple of decades raised this number of transitor equivalent circuits first to hundreds, and then thousands, and tens of thousands. Eventually we had the equivalent of a whole computer on a chip and memories had likewise been so condensed that some small desktop computers had power and access to memories far beyond the capacities of the largest computers available at the beginning of the working careers of most of the older professionals working in the industry.

Many types of new hardware and circuits were developed during this period and manufacturers of hardware are still developing competing technologies. Unfortunately, because of space requirements, a discussion of those various hardware technologies such as a gallium arsenide, MOS, CMOS, Josephson juction etc. will have to be left to another occasion.

The key point here is that, while there has been a revolution in the hardware component side of computers over the last few decades, there has been little change in the techniques and architecture of the software side. It is in this area that most researcher in A.I. (Artificial Intelligence) feel that major discoveries will have to be made if the ambiitious goals of the field are to be obtained.

The prevalent Von Neuman architecture in computers works in the following manner. There exists in the computer a two stage clock that is either in an instruction phase or a data phase. In other words, according to the clock, the computer is either to be getting an instrucion or executing that instrucion upon a particular set of data. The computer tick-tocks back and forth between getting the next instrucion and executing the instrucion many times a second. In fact, in the largest computers millions (MIPS - Millions of

Instructions per Second) or even billions (BIPS) of instructions per second.

Consequently Von Neuman machines (most existing computers) are what are called SISD (Single Instrucion Single Data) machines. Looking below you can see that there are three obvious alternatives. Each of these will be discussed in the next four section.

Thus we have:
SISD - Single Instruction Single Data
SIMD - Single Instruction Multiple Data
MISD - Multiple Instruction Single Data
MIMD - Multiple Instruction Multiple Data
In the fifth section we describe present and operational MIMD systems with a special interest in the ZMOB processor (Maryland machine). Section 6 is devoted to the known supercomputer performance and computer physics. Section 7 gives a review on new computer architecture and parallel processing with connecion to a data structure of algorithms. In section 8 we discuss a special purpose machine in connection to arrtificial intelligence program.

I COMPUTER ARCHITECTURE ON SISD MACHINES

The classical von Neuman computer is a single instruction single data machine (SISD). IBM Amdahl, and Univac mainframes as well as all mini and microcomputers fall into this category. Essentially a single operation is performed on a single datum and the speed of operation is a linear function of logic speed and a memory access time. Although another order of magnitude increase in speed appears likely over the next decade, further improvements will be incremental at best. A number of mainframe manufactures offer multi-processor systems: these should not be confused with parallel processors. A multi-processor can execute a number of separate programs simultaneously, each on a separate processor. In a parallel processor the individual processors collaborate to execute a single program.

2. COMPUTER ARCHITECTURE ON SIMD MACHINES

Single instrction multi-data (SIMD) machines (also known as vector processors or array processors) execute the same instruction simultaneously on many items of data . For example, in principle, a powerful array processor could square the value of each of the tens of thousands of pixels that comprise an image. It is not unusual for super computer manufacturers such as the Cray and the Cyber to include an auxiliary array processor and to quote its processing rate as being their performance rate. Since the processing rate is the product of the arthimetic speed and the number of data elements it is often in the range of tens of megaflops (megaflop-a million floating point operation) per second. Unfortunately only a relatively small subset of problems are amenable to vectorization. Although some of these problems are of central significance (ie. matrix inversion) in physics and computer graphics, they are not particularly applicable to A. I.

In the past few years the so called Systolic Array architecture has made possible a vast increase in the speed and versatility of array processors. As yet no commercially available systems contain systolic arrays. It is our opinion that an advanced super_computer facility should incorporate a systolic array processor ([1], [2]). A single instruction multiple data might be arranged in the following manner (see Fig. 1). From the instruction store each processor A through E would be loaded with the same instruction at the same time and on the next clock moment each would process a different data stream. These could, for example, be the payroll record for different individuals. Each individual would have their hours computed at the same moment, and then their first deduction at the next moment, and then the next, and so forth so that five individuals' payroll records would be computed simultaneously. Theoretically this would be five times faster than using the Von Neuman Architecture. In actuality, because each employee record would not need the same treatment and because of problems in organizing the

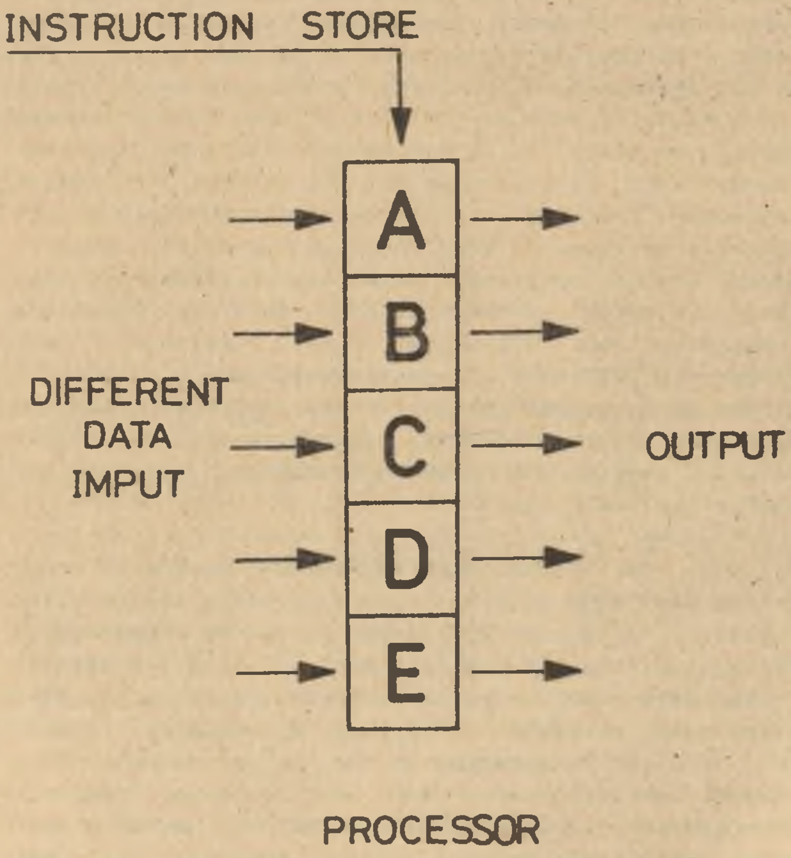

Fig. 1. SIMD architecture

data and using auxiliery hardware such as printers, these exact efficencies would not be realized, but the example demonstrates the principle. Moreover, it should be noted that most multi-processors architectures do not envision the use of just five processors but are usually designed around 64, 128, 256 or even larger multiples of processors.

3. COMPUTER ARCHITECTURE ON MISD MACHINES

Multi instruction single data (MISD) machines are also known as pipeline processors. Such a system may be conceived of as a single datum which is operated upon successively in a series of arithmetic processing stations. Super computers such as the Cray and the Cyber achieve their high processing rates by employing a large numer of these processing stations and thus avoiding the store and fetch operations which precede and follow each arithmetic operation in more conventional architectures.

Here, as shown in the following diagram, (Fig. 2) each processor has a different instruction. All the processors work at the same clock moment, the data is passed from one processor to the next. Since, theoretically, the processor does not have to get a new instruction at alternate clock moments but each time only gets a new piece of data, by this factor alone, the architecture is twice as fast as the Von Neuman architecture. Once again there is added to this advantage the additional multiplication of speed-up of whatever actual number of processors are used (see Fig. 2).

This method of processing is called pipelining because the whole employee record passes from processor to processor as if it were going down an assembly line with each processor performing its single function upon it. It takes the pipeline a moment to fill up, but this can be a small disadvantage. In actuality, a payroll is not a particularly good example in this case because each employee's record consists of a number of distinct parts that require different types of processing. There are other types of problems where numerous operations are performed upon . the same type of data one after the other, and it is in these types of operations the array processors of this design have been favourably

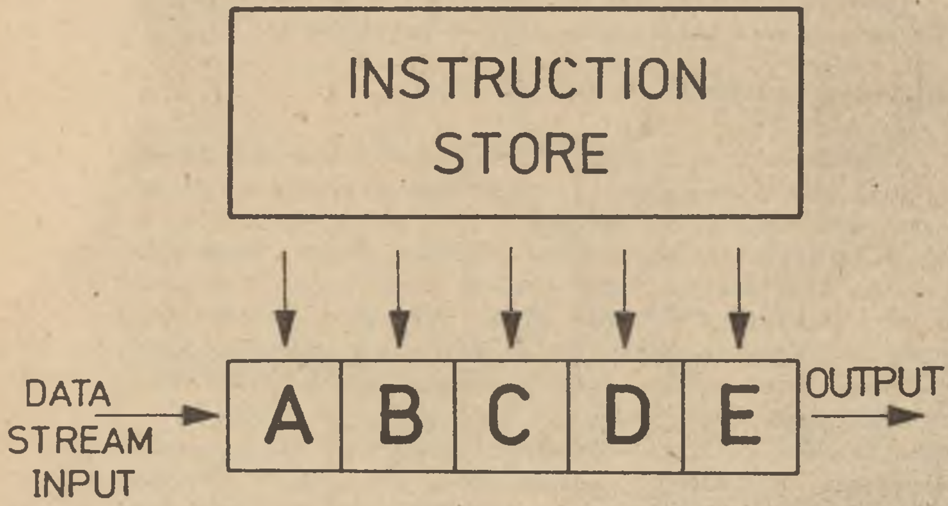

Fig. 2. MISD architecture

applied. Examples are mathematical array processing, weather, and economic modeling.

4. COMPUTER ARCHITECTURE ON MIMD MACHINES

Multi instruction multi data (MIMD) machines are true parallel processors. All the processors cooperate in solving the problem. Ideally such systems would be transparent to the programmer and the assignment of processors and the inner communication between them would be performed by the systems compiler and operating system.

It should be emphasized that such multi-processors and compilers are at present experimental, inefficient and fragile. Other than assiging "do" loops to separate processors, performing in depth searches and explicity designated parallel portions of the program (these are by no means easy tasks) until recently little progress has been made.

In A. I. a number of interesting theoretical proposals have been made. A recent paper by Hiroshi Nakogawa entitled "Parallel Prolog with Diveded Assertion Set" represents a promsing approach. In applied mathematics another recent paper by Pan and Reif at the Courant Institute suggests a parallel processing approach to the solution of large systems of linear equations. This approach if successfully implemented has enormous commercial possibilities and would provide a powerful tool in many areas of physics and engineering.

The most desired type of desgn is a Multiple Instruction Multiple Data type of configuration. This type of architecture envisions there being some large multiple of processors interconnected in perhaps every way possible. The difficulty is largely in designing the interconnenction bus. This is in itself a subject of architecture design of such scope that it will also have to be left to another opportunity. Moreover, there is a second and equally large problem with MIMD computers and that is development of a satisfactory software language. That is to say, a language that would be reiable, comperhensible, and efficient for use by large numbers of programmers. No really satisfactory solutions have been found to either of these two major problems but great numbers of researchers are working on them throughout the world.

The reason that so many researchers are interedsted in the MIMD machine are several-fold.
1) The obvious power of such configuration
2) Its ability to simulate any of the lesser configurations.
3) That A.I. researchers often theorize that there is some similarity between such a configuration and the human brain.
4) That it would be useful in machine vision, and therefore for pattern recognition, and perhaps perception if such a thing is theoretically possible.

It is this last application that is of the most immediate interest to us although there are many auxiliary aspects of A.I. to which it could be applied, such as Robotic Languages, which are almost of equal interest. The reason an MIMD machine could be of such value in machine vision is that there could be many sensing points that could be processed simultaneously. This appears in fact to be the way that the human eyeball is wired. Whereas at the moment, using Von Neuman architecture, the procedure is to scan the surface of a device such as CRT (Cathode Ray Tube)point by point and line by line at a time. The MIMD configuration could allow many points of input to be processed simultaneously and therefore make available in real time a complete image for pattern matching and processing (Fig. 3).

5. PRESENT AND OPERATIONAL MIMD SYSTEMS

Although business magazines have in recent months published a plethora of articles on parallel processing (Business Week) and the IEEE Special Inerest Group on Computer Architectures has for many years presented conceptual designs for parallel processors, very few systems have ever been built, fewer still have become operational, and only four are commercially available.

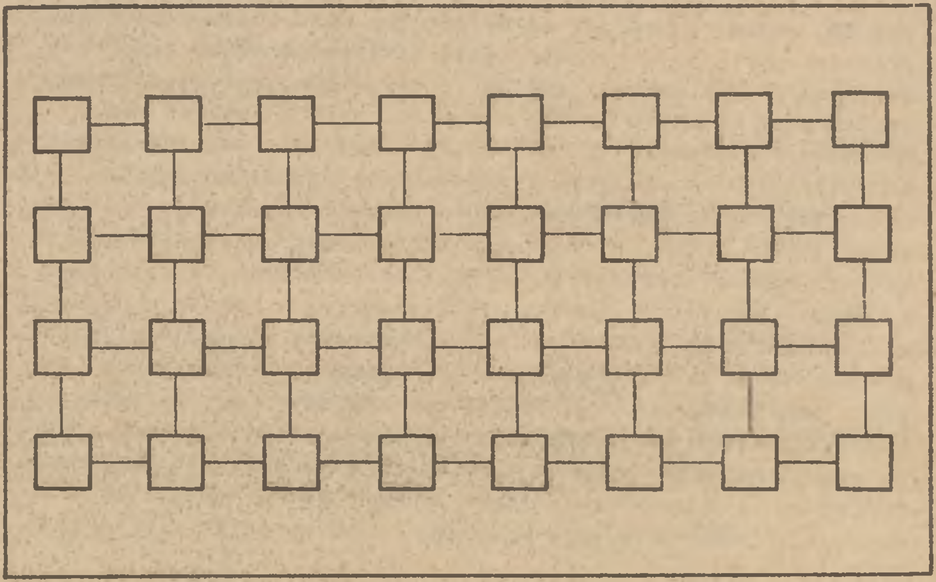

Fig. 3. MIMO example. Some multiple number of processors interlinked through some interconnected bus arrangement in a machine vision application (for example). there could be 1024 of the processors with each one being linked to a single point of some optical devices

a) The Bolt Beranek and Newnam Butterfly.

The BB and N butterfly is by far the most popular parallel processor in the world. At least sixteen machines have been sold and about thirty are on order. Each processor node consists of a 68000 micro computer with a 1MB of memory. The memory of every computer is accessible to any other computer with a miniscule access time penalty through a cross bar switching arrangement. Each node has a cost of about $10K (U.S.). A system with 512 processors, secondary storage backup, printers, terminals and a front system such as micro Vax would have a budgetary price of about seven million dollars (Canadian). The butterfly is an elegant architecture. It enables a large number of processors to work independently on different aspects of a problem employing different data sets and is able at the same time to simultaneously work on a single data set. It has been argued by its proponents that a variety of data flow and control flow architecture can readily be embodied within the BB and N fabric.

b) The Denelcor Nexus.

Denelcor has sold at least five multi processor systems to prestigious computational research institutes such as the Los Almos Institute for Non Linear-Studies, the US Army's Ballistics Research Centre, Laurence Livermore Labs, Messerschmidt A.G. as well as to the usual unnamed U.S. government agencies. The Denelcor system consists of four very high speed ECL processors each equipped with an ECL memory and interconnected with a high speed switch. The quoted price for the system, including a front and processor, is about seven million dollars (U.S.). Denelcor argues that each of its pipelined processors can sustain 32 independent processors and the time of execution of 128 programs compares more than favourably with the BB and N system. It would seem that very few have been convinced by their argument. In our opinion the Denelcor system is a

brute-force attempt to achieve performance through the use of very high speed logic.
Although many processes can be supported, the extent to which processes can be compled or can communicate with each other is limited.

c) The Intel Cosmic Cube.

The Cosmic Cube was developed at the Berkeley Campus of the University of California and applied to a variety of problem areas in theoretical physics. Intel Corporation began to produce commercial systems in mid 1985 and about eight have been delivered. In the Cosmic Cube each 8080 processor can communicate with six adjacent processors in the large class of problems which are susceptable to relaxation techniques. The Cosmic Cube functions very effectively and simulation in the area of quantum chromodynamics on a lattice have yielded a number of valuable and even profound insights. A sixteen node system has price of about $400, 000 (U. S.)

d) Thinking Machines Inc. Connection Machine (TMI).

TMI's Connection machine evolved from the doctoral dissertation of its founder Danny Hillis. It consists of 64. 000 single bit processors each of which has a 500B memory. Each of the processors is associated in hyper cube with six of its neighbours. Effectively the Connection machine is a gigantic Cosmic Cube whose interconnection pattern can be arbitararily defined. It is designed to remove the so called semantic gap which exists between higher level computer languages such as Lisp and Prolog and the basic hardware fabric of a computer. For example, a semantic network can be effectively embedded and directly manipulated in the Connection Machine.

Although Hillis and his colleagues claim they have exceeded the Cosmic Cube benchamarks, others have argued that the Connection Machine is only an associative memory under the control of a symbolic Lisp processor. Others have suggessted that it is a single instruction multiple data (SIMD) system and as such is restricted in the class of problems to which it can be applied.

The Connection Machine is stricty connected to the Connectionist Hypothesis. Thus we add some remarks on this subject. Artificial Intelligence began in the early fifties as a program of cooperation between engineers, mathematicians and neuro-physiologists to understand and simulate the higher functions of the central nervous system. By the late sixties the enthusiasm generated by the seemingly remarkable properties of the Perception had dissipated and the central thrust of AI research shifted abruptly to knowledge representation, that is to say techniques for encoding knowledge so that it could be accessed and manipulated by computers. By the 1980's techniques for representing knowledge in semantic networs or as logic expressions in the first order predicate calculus, had evolved sgnificantly but the process of manipulating these representation presented almost insuperable obstacles to conventional computing techniques. The Japanese Fifth Geneeration program is to large extent an attempt to solve this problem with radically new computer architectures.

At the same time a new school was developing among AI theorists who returned to the AI position of the fifties. Central to what has come to be called the Connectionist thesis was the observation that cognitive tasks such as recognition occur in about 300 msec and that a synaptic transmission requires about 3msec, so that only about one hundred operations can be executed to perform a complex task. In addition the amount of information transferred between synapses must be relatively small. From these facts the Connectionists argue that the massive programs common in A. I. are qualitatively different than the mechanism that governs mental functions and that the structure and interconnections in the system that is carrying out the computation are paramount. Recently David Touretsky and Geoffrey Hinton of Carnegie-Mellon University in a paper entitled "Symbols Among the Neurons: Details of a Connectionist Inference Architecture" put forth a program which could provide a bridge between the connectionist hypothesis and current research in logic programming.

A motif which repeatedly appears in all A. I. discussions is the need for highly parallel architectures to simulate the massive amount of interconnection in the brain, to perform pattern matching, unification, and breadth and depth searches. We do not wish to imply that parallel architectures will solve all the problems of A. I. Extensive research in non-monotonic logics, default logic, modal logic and many other areas of formal logic is essential if progress is to be made. However powerful parallel processors are as important to progress in AI as telescopes and microscopes are to progress in astronomy and bacteriology.

e) The University of Maryland ZMOB Processor

The ZMOB is certainly the most venerable of the multiprocessor architectures. The design originated from a dissertation by Chuck Rieger and was implemented with funds from the Air Office for Scientific Research (AFOSR) in 1980. About three years were required for the system to become operational with 64 processors. Since that time it has been employed for research in areas related to computer vision and music, artificial intelligence, intelligent data bases and numerical analysis.

The system employs Zylog Z80 computers each with 16KB of memory and 1KB of ROM. The processors communicate with each other by means of very versatile bus. The entire system can be visuallized as a circular conveyor belt with a number of stops and a processor at each stop. As instructions and information circulate around the bus each processor can accept or reject data. This bus structure enables any processor to broadcast instructions and data to any other processor to any subset of processors, or to all processors. In addition, a pattern matching function is available which enables the processors to function in an associative manner.

At present a new advanced version of the ZMOB is in the process of implementation under the auspices of the U. S. National Sciences Foundation. The new machine will utilize a refined version of the bus structure and powerful 68000 processors. Since the software is for the most part written in C no major difficulties are anticipated in porting it to the new system.

Although the ZMOB architecture is inherently slower than the BBN Butterfly it is our considered opinion that the architecture is more flexible and that it provides an excellent framework for research in parallel architecture. In addition the opportunity of collaborating with a prestigious centre for computer research, whose work is in the public domain, is an asset of considerable value.

The following is a description of this architecture. The machine consists of 256 autonomous processors (Z80 8-bit microcomputers) each with 64K bytes of memory connected together by a ring-shaped high-speed communications system called the "conveyor belt".

This conveyor belt conceptually consists of 257 rotating mail bins which, at any given point in time, are each under 257 "mail stops". Each of these mail stops, turn, is associated with Z80 processor, except the 257th, which is used for communications with the ZMOB external host computer. The conveyor belt is implementd by a 48-bit-wide 10 MHZ circular shift register, where each stage of the shift register represents a mail stop. Each processor contains, as well as its conveyor belt interface, a high-speed parallel and serial interface to the "out side world" for special applications. A special processor occupies the 257th conveyor belt slot that interfaces ZMOB to DEC Unibus connectd to a host VAX11/780.

In summary, the processor is an autonomous 10MHZ Z80 system including 63K of 357μs RAM and 1K of 375μS EPROM with single-bit parity detect, eight-level vectored priority interrupt logic, 8-bit by 8-bit integer multiply and 32-bit floatingpoint (AMD9511) hardware, 19.2K band synchronous/asynchronous external serial and 24-bit high-speed parallel interface, and the interface logic for its mail stop. ZMOB as whole consists of 256 processors sharing a 16-million-byte memory that is capable of executing approximately 100 milion instructions/s, and communicating via a conveyor belt that switches messages at a rate of 20 million bytes/s. In order to facilitate convenient communication. ZMOB's conveyor belt has four addressing modes. These are "send to processors by address", "send to processor by pattern", "send to all processors" and "send to all processors by pattern". A message always contains the address of the sender and information describing the intended receiver(s) (the destination field), as well as data. In turn, each mail stop contains a unique address (0-256) that can be used to identify it when sending or receiving and hardware enabling it to recognize a specific pattern in messages destination fields. In "send to processor by address", the message sent is intended only for the processor whose unique address matches that of the destination field, in "send to processor by pattern", the message is intended for the first processor whose specific ("posted") pattern matches that of the destination field, in "send to all processors", the message is intended for all processors (regardless of what is in the destination field). Lastly, in "send to set of processors by pattern", the message is intended for all processors whose posted pattern matches that of the destination field.

Each message is sent under exactly one of these modes. In the first two modes, there is only one intended receiver: upon receipt, the mail stop will "consume " the message. In the second two, the mail stop will allow the message to continue after copying it: it is incumbent on the sending processor to intercept its own message after it completes an entire conveyor belt revolution.

In addition to these four (sender-specified) modes, the receiver may futher condition its mailstop to accept messages only from a specific processor,. the "exclusive source" mode. This mode allows an uninterrupted flow of messages between two, or one and a group of processors. It also allows overhead-free establishment of communication, should the sending processors's address be known a priori (and the receiving processor's exclusive source address be set to it beforehand) since, to the processors. It seems as if they have exclusive access to the communication belt.

Reception of messages by pattern is useful in systems where the specific processors that may respond are not known beforehand, e.g., is implemented via a PATTERN/MUSK register pair in the mail stop. One bits in the MASK register represents DON'T CARE position in the PATTERN, and zero bits represents the opposite. A successful match occurs when the CARE positions of the PATTERN register match the message's destination field. The conveyor belt is organized to arrange for optimum efficiency in simultaneous communication between processors. Each mail stop buffers one message in the outbound direction, and can buffer one message in the inbound direction (from the belt). Each processor owns exactly one bin in the conveyor belt through which it can send one message every belt turn. One 48-bit bin passes under each mail stop every 100 μs, under control of a careffully synchtronized 10MHZ master clock. When a processor's bin arrives back at the processor's mail stop - as it does simultaneously for all processors and is signalled by a special "index pulse" from the master clock --if the bin is empty and the mail stop's outbound buffer is full, the message is injected into the belt stream and a special processor interrupt is generated (indicating that the outbound message has departed and the buffer is prepared for another). Otherwise, speed pattern matching logic attempts to recognize the destination field of the message (if any) in the newly-arriving bin.

If the bin is full, the pattern matcher's circuitry deems the message appropriate (including with regard to any exclusive source addres), and if the inbound buffer is empty, the message will be read from the belt and a second type of processor interrupt will be generated (indicating that a message is available to be processed, if only to remove it form the mail stop buffer so that another one may be received). If the addressing mode of the message indicates that the message is also intended for other processors, the mail stop allows the message to continue in its bin; otherwise, the mail stop empties the bin and consumes the message. Messages on the conveyor belt intended for a processors that, upon arriving, can not be accepted by its mail stop (the inbound buffer is still full, the exlusive source addres in effect does not match that of the message or the pattern matching circuitry is simply turned off) will continue to circulate until they are eventually received and consumed, or until the sending processor "reads back" the message (as for multiple-destination addressing modes) and removes it. This apples to a sending processor whose outbound buffer is full, finding that the last sent message is still circulating in its bin in that it must wait until any one of the above conditions is fulfilled before it can send the message.

The conveyour belt operates at such high speed relative to the processor's bin makes a complete revolution in 25.7 μs, the time to pass through 257 mail stops at a 10 MHZ rate, and thus a processor can send one message packet to one or more other processors every 25.7 μs. The 48 bits of the message are further sub-divided into four fields: an 8-bit control field, a 12-bit source field, a 12-bit destitnation field, and a 16-bit date field. Four of the control field bits can also be used as data, yielding a total of 20 data bits. Hence, the actuaal data rate of the conveyor belt is 10.3 μs/byte or 97220 bytes/s. If communication is limited so that, at most, one message can arrive per belt turn (easily effected by setting the exclusive source address mode), it may barely be possible for a processor to send one message (load a message into outbound buffer from a pre-designated location) and receive one message (take a message from the inbound buffer and store it in a pre-designated location) without noticing any effects due to the communication system (that is having to pause because it did not load its outbound buffer in time for the index pulse or because its outgoing message was not picked up in time by the receiving processor). At worst, this could be the case by slightly alterning the conveyor belt speed, e.g., by slowing the master clock or inserting additional processors and mail stops into the belt (it would be unfortunate to have to compromise some of the system in order to make it practical because of the contemporary limits of multiprocessor technology).

The special 257th mail stop which connects ZMOB to its external host VAX 11/780 via Unibus has two special control privileges. The first is that it can write into any of the conveyor belt bins, allowing rapid loading and unloading of data to or from all processors simultaneously; in fact, it is so fast (256 messages/revolution or 25 Mbutes/s) that the Unibus transfer rate (2.5 Mbytes/s) is the limiting step in the process, The second is that it can send special "control" messages (adress in the normal fashion) which have access to processors' mail stops guaranteed by being able to bypass any of the mail stop's inbound buffer. This allows absolute access to any or all processors form the external conveyor belt messages. Also, it allows absolute control over any or all processors by being able to generate non-maskable Z80A interrupts which can bring a processor back to its (EPROM-based) resident kernel operating system.

Software for ZMOB can be any of a number of lanuages written to run on the Z80 under the CP/M operating system. Specifically, these include C, a general purpose language with both high and low-level characteristics; Lisp, a mainstay for artificial intelligence research; and Micro-Prolog, a Z80 version of the theorem proving and logic-based applications language Prolog.

Programming on ZMOB is facilitated by a number of tools that run on the VAX 11/780, including a cross compiler for C, a cross assembler for Z80, and a simulator written in VAX assembler for efficiency.

The ZMOB can make an efficient use of its parallelism, and as a result, should have substantial speed advantage in many image processing situations. In particular, it is worth to outline efficient ZMOB communication/computation schemes for point and local operations (with particular reference to how the data should be partitioned among the processors), discret transforms, geometric operations (in some cases) and computation statistics. These schemes demonstrate that efficient use of ZMOB's parallelism is possible for essentially all basic image processing and analysis tasks. In all of these mentioned tasks, ZMOB achieves a speed-up over a single processor's performance by a factor approaching 256, the number of processors. ZMOB, with its ansynchronous, geometry-independent nature, is a particularly appropriate machine on which to implement a wide variety of object-oriented software. This popular programing style in turn has applications in many diverse areas, including computer vision and image processing, VLSI design, causal monitoring, natural language parsing and simulation. A case in point is the domain of simulation of mechanisms. Here not only do processors assigned to parts communimicate by means of message-passing to effect a simulation of motion, but in addition are able to use the object-oriented system design to gain efficiency when displaying this motion. This is accomplished using various parallelizable graphics algorithms.

It is woth mentioning the advantages of ZMOB architecture for operations on strings. Many operations on strings of lenghth N can be speed up by a factor of P using P processors. String operations can also be sped up even when a single processor is used, by compactly encoding the strings, e.g., using new lenght code. This is very important in image processing if we combine these two ideas. It is well-known that the best way to distribute an nxn picture specified by its pixels grey levels is to partition the picture into P sub-pictures of size N x N each. Each processor can be responsible for one sub-picture.

If a picture is represented by the new lenght codes of its rows, operations such as finding the number of black pixels in a picture, or taking the AND or OR of two pictures can be done row by row. This is possible because the twodimensional properties of pictures are not used in these operations.

ZMOB seems to be a very effective processor for parallel searching and merging. In particular, a speed-up of log (P) is possible for the problem of finding an element in N-element sorted list, and speed-ups of P/log(logP) and P are possible for merging N-element sorted lists on P processors in cases when N = P and P< N, respectively.

In practise, these speed-ups are not attainble, since the shared memory models ignore many practical considerations in multi-processor systems such as interprocessor communications, distribution of data on local memories and limited fanout of memory locations. Taking into consideration existing ZMOB architecture with its communication facilities, it is possible to show that there are:

1. O(log N/log P) algorithm for searching an N-element sorted list distributed on P processors,

2. O(N/P) algorithm for merging two N-element lists on 2P processors,

3. O (log N) algorithm for merging two N-element lists on 2N processors.

Simultaneouly, it is worth mentioning that the lower bound for merging two N-elements lists on 2N processors is O (log [log N]).

It is possible to extend ZMOB architecture to two-dimentsional "conveyor belt-like" configuration (reported by G Heil). In this case a speed-up factor could be squared.

6. SUPERCOMPUTER PERFORMANCE AND COMPUTER PHYSICS.

In this section we give a review of the performance of known supercomputers and some remarks on computer physics.

Computer	Compiler	MLOPS	Maximum theoretical Speed MLOPS
CRAYX-MP-1	CFT (Coded)	33	1600
CDC Cyber 205	FTN	25	800
CRAY-IS	CFT (Coded)	23	160
Fujistsu VP-100	Fortran 77	19	250
Hitachi S-810/20	FORT 77/HAP	17	800
CRAY IS	CFT (Rolled BLAS)	12	160

The computer performance has been tested in Fortran environment using standard linear equation software in full precision artithmetic (64 bit arith).

The above computers have the classical architecture with some improvement to parallel architecture.

Parallel Computers

Computer	Maximum Speed
MPP	6.5 BIPS
Connection Machine	10 BIPS
Non Von	16 BIPS
IPSC Intel	2-8 MF LOPS
Butterfly	200 MIPS
Sigma-1	100 MFLPOS
Cedar	10 MFLOPS

MPP = Massive Parallel Processor
BIPS = Billion instructions per second
MIPS = Million instructions per second

Sigma-1, deleloped by Japan's National Laboratory, will start to work in 1987. Non Von is pure theoretical construction.

The remaining machines are operational. However, it is impossible to test their performance in a similar manner as for classical supercomputers, because the software has not been develped.

The connection machine, developed be the Thinking Machine Corp., seems to be very promissing solution in the computer architecture. It is very flexible. The host computer can change connections according to the nature of the problem to be solved. Probably, it is necessary to develop new parallel/data flow programming languages in order to get the full power of this machine and to synchronize it with the

host computer (Vax, or maybe one of the classical supercomputers).
The high speed of the classical supercomputer has been achieved basically due to the very short clock pulse (high frequency) and due partially to parallel processing. The new full parallel/data flow supercomputers can get the high speed of operation due to new architecture. There are physical limits on the high frequencies imposed by laws of quantum mechanics and velocity of ligth. The limit of the high freequency will probably be saturated in 1990, even if superconducting devices (Josephosn's junction) will be applied. I do not see any limits for a new type of architecture (exept limits on human creativity). The most interesting point of view for a future designer is an interplay between a new architecture and new physics applied as a material realization. Maybe a new branch of physics computer physics — will cure this problem ([3], [4], [5]).

The speed of light and Heisenberg uncertainty priciple limit the processor in the following way. First of all let us consider the processor with a period T of its central master clock. If the frequency of the clock is suffitiently high it will limit the size of the processor. For example if v = 1G Hz (period of order 1 ns), during one period the light travels 30 cm. In general we have a condition

$$l \ll \frac{c}{v} = cT$$

(were l is a size of the processor, and c a velocity of light) in order to make a master clock time global for all processes in our computer. This makes as to pack very closely all the elements of hardware i.e logic gates, flip-flops etc. However this will cause mamy problems. First of all we should remember that a computational process is a physical process and it dissipates an energy. For this in the case of very high density of logic elements the dissipation of a heat can destroy the processor. This forces us to go to molecular, atomic or maybe even nuclear processes as a material realization of the logic. In this case the laws of quantum mechanics start to play i.e.

Heisenberg, uncertainty, principle. But not only, we should also change our philosophy of computer (logic) design. This new ideology of a design has been introduced by R. Feynman in 1985 ([3]). Let us describe it briefly. Let a vacuum state $|0\rangle$ corresponds to the logic value 0 and the first exited state $|1\rangle$ to the logical value 1. Let us introduce operators of creation and annihilation a^{+}, a for this state. Now we can express any, logic function using these operators, for example NOT, NOR, NAND, AND, OR, EXOR etc. We can also express more complicated logical divises as half-adder, adder, multiplier, shift register, flip-flop etc. We can also introduce a timing via an evolution operator $U(t)$ connected to the hamiltonian of the devise $U=e^{iHt}$ Defining the so called "ballistic computation" we can find this hamiltonian and desribe in terms of spin-like waves (as R. Feynman). We can also proceede in a diffterent way working with a periodic time-depending hamitonian. Let us come back to the limitation of this system. Let an energy gap between $|0\rangle$ and $|1\rangle$ be E and the width of an energy level Γ. The Heisenberg uncertainty principle limites the minimal switch time t_{min} of any logical device: $(t\cdot\Gamma\geqslant\hbar)$

$$t \geqslant t_{min} \geqslant \frac{2\hbar}{E} \quad , \qquad E \geqslant 2\Gamma$$

Thus we see that in the case of 1eV (spin wave) we have $t_{min} \sim 10^{-15}$s For 1 keV, 1MeV one easily gets 10^{-17}s, 10^{-21}s. This will force us to consider nuclear or elementary particle (high energy) processes in the last case. From the other side we will probably change a theoretical model of computation i.e to consider Quantum Turing Machine in a place of ordinary Turing Machine ([4]). This goes to a very interesting considerations on quantum parallel computation and a simulation of quantum processes. For example we can get a real random number generator. The problem of implementation of parallel or concurrent architecture in quantum hardware seems to be very promissing. Some researchess in Watson's IBM Research Centre are very enthusiastic for such innovations ([5]). The interesting point is to consider new type of algorithms and languages designed for this type of computers. The speculations on

A. I. program implementation in such machines seem to be also interesting. This is strictly connected to, the quantum indeterminism in the machine. Some ressearchers in A. I. program claim that the real intelligence can emerge only if we have some kind of indeterminism. This indeterminism is intristically probabilistic and has nothing to do with a classical nondeterministic Turing machine.

7. NEW COMPUTER ARCHITECTURES, A PERSPECTIVE.

The von Newman-type computer, invented by John von Newman, is a sequential machine. It means that every instruction has to be executed step by step. This is reasonable, because we need, in general, a result of a previous operation in order to execute the next operation. Moreover, there are many algorithms for which this statement is not terribly important. The most important exaples are as follows:

1. matrix multiplication
2. matrix addition and substraction
3. matrix inversion
4. inner product calculation
5. quick sort
6. tournament sort
7. external sort
8. FFT (Fast Fourier Transform)
9. convolution

All of these algorithms have a common property: some of the manipulations on data can be done in parallel. We do not need the results of one operation in order to execute the next one. Moreover, the traditional von Newman-like architecture forces us to proceed with the computations (manipulations, transformations) in a sequential way. It is natural and important to make an effort to design a computer architecture which will allow computations according to the algorithm structure. In this way, the gap between an algorithm and a machine would be much smaller and, due to this, the computation process would be quicker. The idea is very simple. From a practical point of view, it is enough to take a quite general and important algorithm and to map it into the processor's structure.

It is necessary to say that I do not mean here an implementation of an algorithm as a special purpose computer. I mean a general purpose architecture based on data structure of an algorithm.

The most important algorithms are: matrix manipulation and sort/search algorithms. They are simultaneously very general from the point of view of an information structure. It means that they can be represented by networks or graphs (trees, binary trees, balanced, almost balanced or tries; the last motion is obtained from the word retrieval). In this way, we do not care what kind of transformation on data has been done in a specific node of a network. It happens that, in the case of a matrix manipulation algorithms, the information structure is a network with periodic properties — an array. In the case of sort/search algorithms, we get rooted graphs (trees). Both structures are very general and , after mapping into the processor's interconnections, we get systolic architecture (systolic arrays) and tree (graph) machines. Some researchers in computer science suggest that systolic arrays and perfect shuffle (quick sort algorithm) should be a source of a new computer architecture for a general-purpose computer (Bayan networks, ω –networks). There are many kinds of systolic arrays: rectangular, hexagonal (Fig.4), 1-, 2- or 3-dimensional. All of them have a common property: the information is flowing in a pulse way (similar to heartbeat). Because of this pulsing property, it is possible to consider the information flow as a wave propagation process that the Hyghens principle is satisfied (every processor is a source of information and this is similar to a wave propagation on the surface of the water). Because of this, some of these processors are called wavefront processors. There are two kinds of architectural solutions for this type of arrays.

In the first, every row of an array is working independently. In the second, there is a synchronization

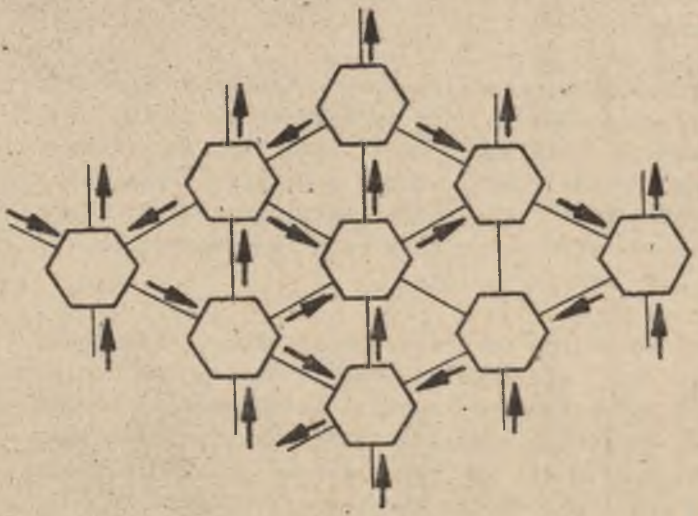

Fig. 4. Hexagonal systolic array for matrix multiplication. Arrows indicate directions of the information flow

between all rows and an array behaves in a holistic way. The first possiblility could behave similarly to a non-deterministic machine (there is not a coherent infomation flow). The second solution has more advantages, because we can consider an information flow as a coherent wave propagation and apply some ideas from wave mechanics up to holography. The wave propagates, of course, in information space (not physical space). The idea of systolic arrays seems to be very attracive, and some people consider data flow and control flow machines. It is interesting to mention that the BBN buterfly computer is bassed on the information structure (interconnection) of the convolution or FFT algorithms.

The tree (trie) machines are also very attractive. However, the information structure is not as regular as in the case of systolic arrays (excluding the case of completely-ballanced binary trees (BBT)). There are also some more complicated architecture, i. e graph machines (Fig. 5). Some of these processors structures have been implemented in the VLSI using MOS and CMOS technology.

Let us conclude that the parallel architecture has been applied by Nature in visual data transformation. Probably, this is the best architecture, because it needs only 10-100 computations in order to proceed visual data in the animal brain. The amount of data is so enormous that everything must be done in parallel. The connectionist approach to A. I. program tries to use this architecture in order to enable intelligent networks to understand natural languages. This is similar to some ideas mentioned in the next section (artificial intelligence program and a special purpose machines) with a connection to Chomsky's linguistics and R. Thom's catastrophe theory. The information structure in the form of a network is typical for neocortex, and this is not accidental (I presume). In this case, a network could be considered (physically), as a complicated Ising-like model. Some researchers in theoretical biology suggest the possibillity of a second order phase transition in such a network (Grodsky's array). On the level of the information structure such a phase transition means a new information channel and a new connection between their elements. Probably, the human brain behaves in this way, because it has been proved that some correlations between functional potentials of neurons have statistical significance. I do not mean any α, β, δ or θ rhythms, which express an average electrical function of the brain. It is now absolutely sure that these rhythms have nothing to do with an information exchange among neurons. For example, in an epilepsy,

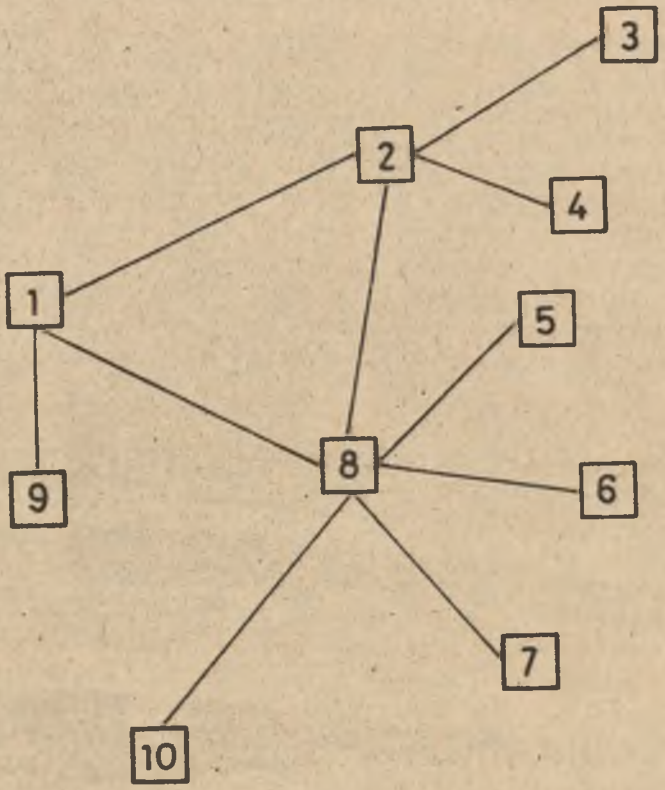

Fig. 5. Graph machine architecture

these average rhythms dominate and, because of this, the normal function of the brain is impossible.

The real function (electrical/physical) of the brain probably consists in correlation of the potential fluctuations between distant neurons (3-4 cm). They are not connected directly; thus, it means there is an information exchange channel between them. This suggests that the connectionist idea supported by phase transition mechanism is quite right. From a different point of view, we know from neuro-pathology that the memory is distributed in a coherent way, similar to the holographic picture. This suggests an information wave propagation mechanism for a neuron network, perhaps similar to wave front processing for systolic-like arrays.

Let us sum up. Any quite general algorithms known in computer science could be a soure of a new computer architecture. Thus, we have many possibillities. There is not, in general, any clear criterium for an efficiency of such a new architecture as a general-purpose computer. If we implement one of these algorithms in VLSI as a piece of hardware, it will work very well as a special-purpose computer for a specific problem solved by this algorithm. However, it could fail for a different problem (for example, it can be very slow, even in comparison to a classical von Newman machine, or it could not work). Supercomputers are considered as general-purpose computers. Because of this, we should choose an adequate algorithm as a map into a processor's structure. This seems to be a question of art, because, in many casese, the architectural improvement does not go to very high efficiency. Moreover, it is very important to look for new solutions on all levels, i.e.:

1. hardware — VLSI system (or even UVLSI)
2. algorithms
3. high-level programming languages,

in order to remove a semanic gap. Semantic gap, roughly speaking, means that there is not a mapping between these three levels, i.e., there is not an isomorphism between the processor's structure, algorithms and information structure of the HLPL (High Level Programming Languages). The perfect solution is to design a new type of HLPL with data flow or control flow mapped into new computer architecture. This is beyond SIMD (or even MIMD) structure applied in Cray-1S, Cray-2, Cray-XMP or Cyber 205. Due to this, supercomputers operate quite slowly. For example, in Fortran environment, they will operate much more slowly for LISP or Prolog (an additional compiler and a higher semantic gap). See, for example section 6. If we do not want to change a programming language (HLPL) because this will cause inconveniennnces and costs, we can implement Fortran, LISP or Prolog in VLSI system in order to get a higher performance. Some of these implementations have been done and we have a LISP machine on a chip (LISP Machine Corp.). The same has been done in the case of Prolog, for MProlog (a hungarian version). In the case of Prolog the systolic architecture has been used. (Logicware Inc.). The japanese Fifth Generation Computer Project uses MProlog as a machine languagge for its Personal Inference Machine. This is, of course, half of the solution. Fortunately, a new high-level programming language has been designed by the Department of Defence in USA. This language is called ADA in favour of Ada Contessa de Lovelace, daughter of Lord Byron, the first computer programmer. (She programmed the steam analytic engine designed in the 19th century, a primitive computer.) This language is parallel and, because of this, more adequate for new architecture. However, it is not popular in the academia and it is not a data flow/control flow language. Thus, we can conclude that we should re-invent a computer (hardware-architecture) programming language, new physical concepts in computer physics, in order to get a real supercomputer. This seems to be very exciting; however, it is very hard because all of these new principles must be in accord.

8. ARTIFICIAL INTELLEGENCE PROGRAM AND A SPECIAL PURPOSE MACHINE.

In general, we can divide the whole artificial intelligence program into two fundamental currents;

1. One which is based on logic programming [6] with connection to modern digital computers:

2. One which is looking for deeper understanding of intelligence due to connection between non-linear networks, thermo-dynamics, catastrophe theory and modern linguistics.

It seems that the second program has many theoretical advantages and probably due to an intriguing relation between Chomsky's lingustic and R. Thom's catastrophe theory, is able to solve an emerging problem of intelligence. This is my personal point of view, but I see this possibility quite clearly.

What is intelligence? ([6]) It is a language able to form models with hierarchal structure and double articulation. The network approach uses the theory of differentail equations and their stability. It looks for limit cycles and attractors. Each such limit cycle corresponds to a "behaviors" of a system. The natural terminology for this problem is the terminology of differential topology, i.e., catastrophe theory which classifies such "behaviours". R. Thom discovered a very interesting relation between the classification of stable (i.e., stable with respect to small smooth deformations) "behaviours" — "regimes", and a derivation tree from modern linguistics (R. Thom — "Topology and linguistics", (in French). In this way, the derivation tree could naturally emerge from the stable configuration of the network. This program seems to be very attractive for the long run, and it is worth studying in more detail.

The special-purpose machine, based on these principles, could probably develop an inteligent behaviour with an ability to learn form experience. Moreover, I suggest choosing the first possibility as a basic principle of a design in order to get some practical results in the future. Why? The second fundamental approach is purely theoretical (up to now) and, to the best of my knowledge, nobody has constructed any robot, any autonomic unit, based on these principles. Morever the second approach has practical application in automated reasoning in CAD, CAM and "intelligent" control systems. Thus, this program offers, in principle, a hope of designing and manufacturing a special-purpose machine for example, a submersible robot which should have the following properties:

1. Controlled form the ship by human staff

2. Able to make correct autonomic decisions concerning the outside situation in a short time;

3. Able to transform information obtained by sensors into data understa—ble by a program

4. Able to translate a logical decision into correct commands for its effectors.

5. Able, in the case of any ambiguity, to ask the control on the ship for a decision.

6. Able to ask a question about an internal or external situation.

These six principles are higly inter-related, and should be considered a minimum requirement for our special-purpose machine.

The fundamental question which immediately arises is as follows: How to achieve it theoretically and what kind of material realization is necessary?

First of all, we will consider the theoretical means. What we need is an effective inference engine which can proceed as follows:

1. Form a question ?Q?
2. Answer this question: A and negate it B=~A
3. Select data D from outside and from inside the vehicle connected to the preposition A.
4. Find all stored knowledge K connected to A and D.
5. Use logical rules and unification algorithms in order to transform the expression B. D. K.
6. If B. D. K. equals 0 (sign of a contradiction), transform into set of commands for effectors and/or communicate with the ship

In order to proceed with this process, we should be able to express A., D., K in a set of clauses, to unify expressions and to apply a set of sound and complete inference rules. K could be stored in an associative memory in terms of clauses. What we need is to translate D into clauses. This could be achieved by a pattern recognition program for outside inofrmation. The inside information could be given in a required form.

Now it is necessary to translate a decision into commands of effectors. This can be achieved by a feedback between a question forming agent and an effector processor which will proceed with its task if the inference engine finds a contradiction. This quite vague program, after its realization, has really nothing to do with an intelligent behaviour, because the model of the situation has not been created. In my opinion, this is the most important in A. I. Moreover, it can be very efficient if we find processors which are able to proceed with all of these manipulations in a very short time. The last problem seems to be very promissing, because of the construction of SUM (Syracuse Unification Machine) on a one chip in CMOC technology.

The question is immediately connected to the material realization of the theoretical problem. If we decided to use digital processors, we have following possibilities:

1. mechanical devices
2. fluidics
3. light devices (optoelectronics)
4. electronic devices (VLSI or UVLSI)

The first two possibilities are inapplicable because of a very long reaction time and huge size. The third possibility is not sufficiently developed (up to now). Thus. it seems that VLSI are the best possibility and we can consider silicon VLSI based on MOS, CMOS, CHMOS technologies. These technologies are progressing rapidly, and we can also condider GaAr technology with MODFET transistors (Modulated Doped Field Effect Transistor). The last technology is very promissing because of a very high speed (up to 10 psec of switching time).

Acknowledgements

I am very grateful to dr A. Stein and dr G. Heil from Epistemics Ltd Toronto Canada for many interesting discussions and advises.
I thank also Mr B. Beach for discussions and a financial support The support of UNCOMMON Foundation is highly apreciated. I want to thank Miss Catherine Lynch for her helpful assistance in a literature search using CAIRS (Computer Aided Information Retrieval Service).

References

1. Kai Hwang and Fayè A Briggs - "Computer Architecture and Parallel Processing" Mc Graw-Hill Book Company, New York 1984.
2. Tutorial on parallel processing ed. by Robert H. Kuhn and David A. Padua IEEE Computer Society, New York 1981.
3. R. Feynman - "Quantum mechanical computers" Foundations of Physics 16p. 507 (1986).
4. D. Deutsch "Quantum theory, Church—Turing thesis and the universal quantum computer" Proceedings of Royal Society of London A400p. 97 (1986).
5. Charles H. Bennett - "The Thermodynamics of Computation - a review" Internatioanl Journal of Theoretical Physics 21p. 905 (1982).
6. Paul R. Cohen and E. Feigenbaum - "The Handbook of Artificial Intelligence" William, Kaufmann Inc. Los Altos Ca. 1982