A PARALLEL ARRAY OF MICROPROCESSORS - AN ALTERNATIVE SOLUTION TO DIFFUSION PROBLEMS

E. BUSTOS, J.D. LAVERS AND K.C. SMITH

DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF TORONTO

ABSTRACT

This paper explores the possibility of using a parallel, special-purpose-processor to solve diffusion problems. The system considered in an N-Dimensional Array of Microprocessors each of which provides an explicit solution of the diffusion equation at its own location. A master-slave architecture is used in which it has been demonstrated that private memory assignment offers a considerable advantage over a common memory array in terms of complexity and the tradeoff between interprocessor interference and memory reduction. For economic reasons all communication in the prototype is by means of serial links. The master-slave communication protocol is very specific with the joint aims of compactness and flexibility demonstrably achieved.

INTRODUCTION

A broad range of physical processes are described by a class of partial differential equations of the parabolic type. Such equations characterize diffusion processes. Typical examples would include the behaviour of temperature with time in any heating process , the diffusion of magnetic fields in conducting media, and the time varying flow of an incompressible fluid. The general form of the PDE involved is [1]:

$$\bar{\nabla} \cdot \frac{1}{a} \bar{\nabla} H(\bar{x},t) = b \frac{\partial H}{\partial t} (\bar{x},t) \qquad (1.1)$$

where: $\bar{x}$ represents an n-dimensional space vector
t represents time
H is the physical variable
a and b are physical constants

All of the conventional numerical methods used for the solution of (1.1) discretize at least the space dimension. In the case of difference solution methods, Taylor's approximation, coupled with the appropriate vector identies, can be applied to the left hand side of (1.1) to yield [2]:

$$\sum_{j=1}^{2D} \frac{\alpha_{i+j}}{a_{i+j}} B_{i+j} - B_{i+j} \sum_{j=1}^{2D} \frac{\alpha_{i+j}}{a_{i+j}} = b \frac{dB_i}{dt} \qquad (1.2)$$

where: D = number of space dimensions
i = the ith point in space
j = identifies the neighbour index (1, 2, 3, 4 for 2 dim.)

$$\alpha_{i+j} = \begin{cases} \dfrac{2}{h_{i+j}(h_{i+j} + h_{i+j+2})} & \text{for j odd} \\ \dfrac{2}{h_{i+j}(h_{i+j} + h_{i+j-2})} & \text{for j even} \end{cases} \qquad (1.3)$$

with h_{i+j} being the distance between node i and i+j.

Given that the LHS of (1.1) can be discretized, either by difference or other methods, one solution method [3],[4] would be to place an analog network at each point of the discretized space so that the electrical response at each node of the network has the form of (1.2). This approach has been successfully used to simulate linear and saturable magnetic field problems for example. Difficulties arise, however, in finding analog devices with suitable nonlinear characteristics.

Alternatively, the RHS of (1.2) can be put into discrete form and a general purpose digital computer (GPDC) can be used to solve the resulting equation, which becomes [2]:

$$\sum_{j=1}^{2D} K_{i+j} B_{i+j}(nT) + B_i(nT) \sum_{j=1}^{2D} 1 - K_{i+j} = B_i((n+1)T) \qquad (1.4)$$

where

$$K_{i+j} = \frac{\alpha_{i+j}}{a_{i+j}} \times \frac{T}{b} \qquad (1.5)$$

and T = time increment

This is clearly an explicit formulation of the problem in which the field value at the new time step can be obtained using entirely old data. It has been shown, however that this solution is numerically stable only for values of K_i such that

$$K_i < \tfrac{1}{2} \qquad (1.6)$$

Although the algorithm is computationally very simple, the stability criterion implies that the time step must be chosen to be quite small, thus resulting in possibly excessive computing time.

As an alternative, time varying PDE's can be discretized in an implicit fashion, thus generating a set of linear equations that must be solved at each time step, by Gaussian elimination for example. Since implicit methods are stable for much larger values of T than are explicit methods, a trade off results. It is clearly between a

CH1465-4/79/0000-0380$00.75

solution with few time steps at each of which the solutions to a set of equations is required, and one with a large number of time steps with a simple algorithm.

While the GPDC approach dominates the field currently, recent advances in LSI technology and multiprocessing point to an attractive alternative for the solution of this type of problem. Such an approach uses a parallel array of microprocessors organized in analogy to an analog processing network.

Some attempts have already been made in this area by Pottle [8] and Hatcher [9]. They propose a type of parallel microcomputer structure designed to efficiently support the solution of a system of linear equations. While such a structure could not be classified as a GPDC, its range of applications is sufficiently great to require a rather complex and expensive machine.

Alternatively, by restricting ourselves to diffusion problems characterized by equation (1.1) we intend to develop an economic parallel processor which combines the properties of simplicity of explicit solution inherent to an analog network, with the accuracy and speed inherent to a digital machine.

DESIGN CONSIDERATIONS

There are three key issues on which the designer of a parallel special-purpose system must concentrate:

a) The general organization of resources best suited for the specific application.
b) The distribution of memory within the system.
c) The communication of data and its corresponding protocols.

Organization of Resources

There are two possible general organizations of resources in a parallel processing system: centralized and decentralized. In the centralized or master-slave structure there is one central controller, the master, in charge of initialization, distribution of tasks, supervision of the process and communication with the external world, and many slave processors, each of which executes a program as instructed by the master and under its control. In the decentralized organization, on the other hand, there is no specific physical controller but rather any of the processing elements (P.E.'s) can take over the function of the master. Thus, in the decentralized organization a P.E. is somewhat more complex than a simple slave function requires.

The choice of a particular structure is determined by three factors, namely the nature of the application, the degree of interaction among the P.E.'s and the efficiency of the system as a function of cost and performance.

For the diffusion problems in which we are interested, the task of each PE consists of the evaluation of equation (1.4). This requires an exchange of information between each element and its immediate neighbours for every time step. Thus the process must be regarded as highly interactive.

The cost of centralized and decentralized organizations for solution of these problems is very generally

$$C = NS_c + M \tag{2.1}$$

$$D = NS_d + E \tag{2.2}$$

where C = cost of a centralized organization
M = cost of a master
D = cost of a decentralized organization
N = number of P.E.'s
S_c = cost of each simple PE in a centralized organization
S_d = cost of each PE able to be both master and slave
E = cost of hardware external to PE's in a decentralized organization

On the basis of actual cost of available components and with the assumption that the design of memory and communication can be made independently, the relationship among E, M, S_c and S_d is estimated to be:

$$M = S_c;\ E = .5M;\ S_d = 1.1S_c$$

Thus, for N greater than 6 the master-slave organization is more economic than a decentralized one. If we add to this result the high interactivity which characterizes solutions of a diffusion process, we may conclude that a master-slave organization is more suitable for this kind of application.

Another issue which involves the organization of resources is concerned with the number of P.E.'s that are necessary. Two more possibilities arise: that there be N P.E.'s, one for each point in the discretized space, or that there be X P.E.'s each assigned to Y points where XY = N.

The cost of each of these options would be:

$$C_1 = NS_1$$

$$C_2 = XS_1 \tag{2.3}$$

where: C_1 = cost of first option
S_1 = cost of each P.E. in the first option
C_2 = cost of second option
S_2 = cost of each P.E. in second option
N, X as defined above

If the remaining architectural decisions are the same for both options, and keeping in mind that the data memory required for each system is the same, we may express S_2 roughly

as:

$$S_2 = S_1 + (Y-1)M_D \qquad (2.4)$$

where: M_D = cost of the data memory required by each point in space

Y = number of points assigned to each P.E. in second option

Thus the percentage of saving attainable with option 2 in relation to option 1 would be:

$$\left(\frac{Y-1}{Y}\right)\left(\frac{S_1 - M_D}{S_1}\right) \quad 100\%$$

On the hand we see that option 1 will be y - times faster than option 2.

Thus it is our conclusion that the increased complexity of each P.E. and the speed degradation resulting from the second alternative, far outweigh the potential reduction in cost that option 2 may represent.

As a result of these deliberations we conclude that a master-slave organization should be adopted in which each slave maps only a single point in space.

Memory Distribution

There are three different poissibilities to be considered in general in assigning memory in a distributed processing system:

a) Establish a main memory common to all the P.E.'s. This distribution has minimum memory redundancy and cost but maximum complexity and interprocessor interference.
b) Retain no common memory but provide each P.E. with its own private memory. Such a scheme exhibits maximum redundancy and cost, but is the simplest in concept with no interprocessor interference.
c) Use a combined scheme with one memory segment common to all or some of the P.E.'s and another segment private to each. Here the complexity is about the same as in the first option but the redundancy, cost and interprocessor interference lie somewhere between that of the extreme options.

In order to motivate the correct choice, figure 1 is included to indicate the possible separation of memory into program and data segments. It was prepared on the basis of considering a large number of code modules for various possible modes of operation.

For the specific problems for which the processor is intended, all the P.E.'s assigned to the same type of job (either inner node or boundary computation)will always execute instructions from the same segment of program memory, but each will work simultaneously with a particular segment of data memory reserved for their associated point in space. Thus to preserve total parallelism, we need as many output and temporary data memory areas as there are P.E.'s. Taking into account that figure 1 indicates that such distinct areas represent 85% of the overall data space required and considering the declining cost of RAM memory, we may conclude that savings that may be obtained by sharing the remaining 15% does not merit the additional complexity and degradation that results.

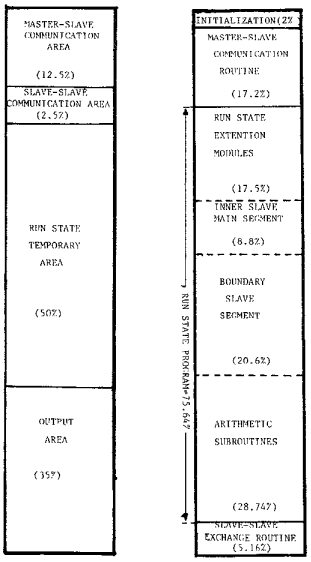


FIGURE 1 - Map of the Memory in the System Showing the Areas Allocated to Different Code Modules; (a) Data Memory; (b) Program Memory

On the other hand shared program memory appears to be a more likely possibility. In view of the code distribution indicated in figure 1 it is apparent that if a significant saving is sought with a common memory scheme, one should concentrate on the RUN segment.

In order to evaluate the degradation of a system with a common RUN segment over one with a private memory, some of the alternatives were simulated on an IBM 370. The simulation was aimed at identifying bounds on the degradation of the system whose architecture is defined in the remainder of this paper, using two of the many possible policies for accessing and executing instructions from common memory as follows:

a) Only one device fetches instructions from the common memory while the other requesting processors remain in a request state.
b) All the processors accessing the same locality in common memory at the same time, fetch and execute the same instruction. When the processors working on the same segment of the program reach a "branch point", the "non-branching" processors keep executing while the "branching" ones wait until they are eventually rejoined to the former.

Figure 2 plots the degradation of computing time suffered for each of these schemes in comparison with a totally separated memory system. Since both curves increase monotonically, it is apparent, for a system with the potential for an unlimited number of processors, that the degradation sooner or later will become unbearable. Thus we conclude that neither of these shared possibilities is suitable. However a third option suggests itself: It is to have a system with clusters of N_1 processors sharing a common memory which is private to each cluster.

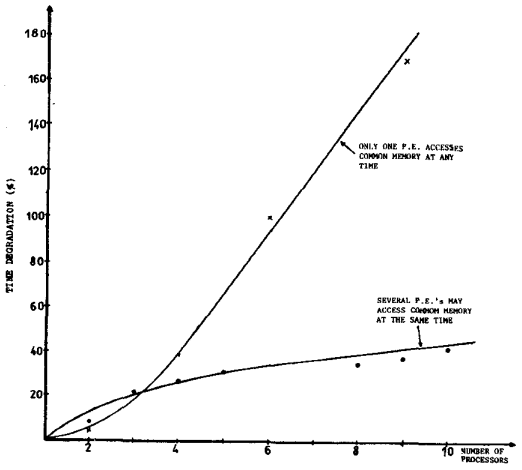


FIGURE 2 - Degradation of the Solution Time of a Common Memory System in Comparison with that of a Private Memory Organization

If we define the cost of a private memory system, P_m, as:

$$P_m = (K_m + K_r)N \qquad (2.3)$$

where:

K_m = cost of the program memory on each P.E.
K_r = cost of the remaining hardware incorporated in each P. E.
N = number of P. E.'s in the system

Then the cost of a system with clusters of N_1 processors sharing a common memory is:

$$C_{pm} = \frac{N}{N_1} K_m + N(K_r + K_e) \qquad (2.4)$$

where N, K_m and K_r are as before and K_e is the cost of the additional hardware required per slave to support the common memory structure.

Thus, the reduction in the average cost per slave attainable with the cluster structure would be:

$$\text{saving/slave} = \frac{(N_1 - 1)}{N1} K_m - K_e \qquad (2.5)$$

Figure 3 plots (2.5) as a function of N_1 with the K_m corresponding to various estimates of the price of program memory. Figure 4 shows the tendency of the prices of semiconductor memory to reduce while figure 5 illustrates the effect of figure 4 on the maximum reduction of cost expected from figure 3.

From a consideration of this set of curves we predict that in the long run the saving obtained with a common memory system does not justify the penalty paid in solution time, in the increase in slave size, and in the loss of modularity.

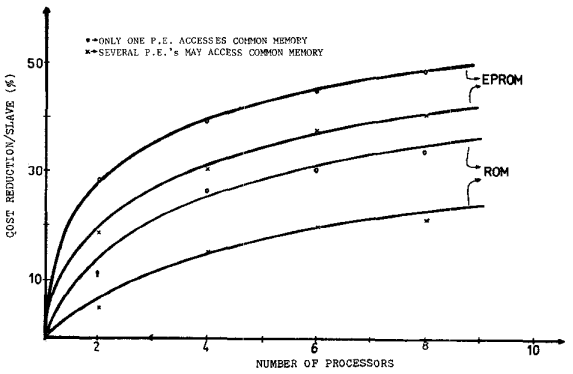


FIGURE 3 - Average Cost Reduction Per Slave Expected with a Common Memory Organization in Comparison to a Private Memory Organization

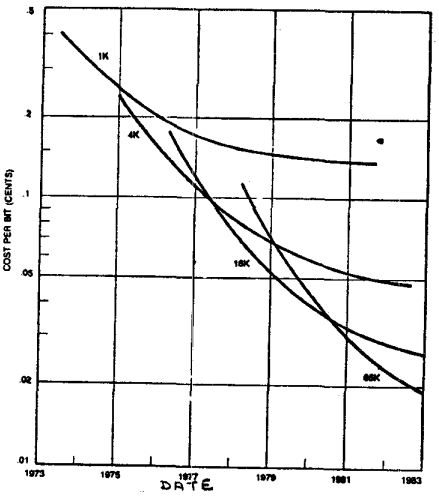


FIGURE 4 - Projections of the Price of Semiconductor Memories

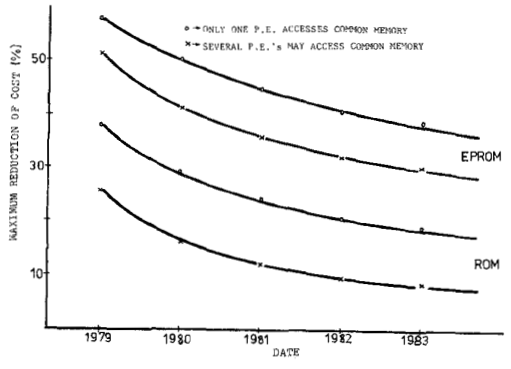


FIGURE 5 - The Decaying Tendency of the Cost Reduction Attainable with a Common Memory Organization as a Result of the Decrease in the Price of Memory

Communication Network

There are two different levels of communication in a distributed processing system such as this: master-slave (M-S) and slave-slave (S-S). For each of these levels there are four decisions to made concerning:

a) The interconnection network as a function of the types of information to be communicated and the number of devices that may take part.
b) The communication protocol corresponding to that network.
c) The type of hardware link.
d) The format of the messages.

Slave-Slave Communication

As has been indicated, each slave requires information from each of its neighbours for each time step. This represents a total of 2ND messages per time step, where D is the number of dimensions of the array and N the number of P.E.'s.

We may distinguish two basic topologies for such communication.

a) Variable topology, in which there is no physical connection among the slaves, but all messages are routed through the master. This configuration, although the most flexible allowing common hardware and software for M-S and S-S communication, becomes inefficient for the solution of this type of diffusion problem because of the large number of "local" messages.
b) Fixed topology, in which a physical channel exists between each slave and its neighbours. Within this category two distinct interface philosophies are available:
   b.1) - whenever a slave requires information from any other it issues a request signal which, when acknowledged by the destination slave, starts the transfer under the control of either of them.
   b.2) - transfer of information takes place in all the slaves simultaneously, with each of them receiving from one particular neighbour "direction" and transmitting to the same neighbour or its "opposite" at the same time. In order to achieve such synchronism, each slave must be halted after its computation for time (t-1) is complete, with transmission allowed to begin when all the slaves have completed.

The last alternative was chosen because it:
- implies the simplest S-S protocol,
- simplifies the supervisory functions implicit in the master role,
- avoids deadlock situations that can arise with the other alternatives,
- introduces the least degradation, and
- does not preclude the possibility of sharing resources with the M-S network.

Figure 6 shows a slice of the resulting system illustrating the communication interface of each P.E. Table I provides a rough estimate of the cost of this interface with either a serial or a parallel link. From it we may see that the former is more advantageous provided that the time for exchange of information can be made insignificant compared with the overall computing time.

Lastly, since all the slaves are identical, use a common clock, execute the same communication routines, and start communicating at the same time, the transfer of data is inherently synchronous.

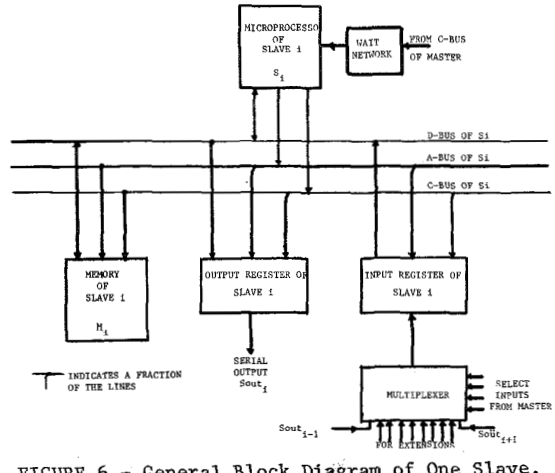


FIGURE 6 - General Block Diagram of One Slave, Illustrating the Slave-Slave Interconnection Network

TABLE I
Estimated Cost of the Communication Interface of Figure 6 with a Serial or Parallel Link

	SERIAL LINK		PARALLEL LINK	
Out Register	1 8-Bit Parallel In-Serial Out	- $1.14	1 8-Bit Latch	- $1.25
In Register	1 8-Bit Serial In-Parallel Out	- $0.68	1 8-Bit Latch	- $1.25
MPX	1 8-Input Multiplexer	- $0.51	8 8-Input Multiplexer	- $8x.51
TOTAL	3 I.C.'s	- $2.33	10 I.C.'s	- $6.58
Lines Out	1 (Serial Out)		8 (Out Register)	
Lines In	8 (Inputs of Multiplexer)		64 (Inputs of Multiplexer)	
TOTAL LINES	9		72	

Master-Slave Communication

According to the nature of the information and the direction in which it flows, we may distinguish five types of messages exchanged between master and slaves.

From master to slave:
a) global data
b) local data
c) global commands
d) local commands

From slave to master
e) local data

Two different interconnection structures allow such an exchange; either direct memory access (DMA) or register interface. Figures 7 and 8 illustrate the M-S interconnection network for each of these structures. Table II contrasts their main features and Table III compares the additional cost and complexity introduced by each.

Considering that the receiving routine associated with a serial register interface represents only 5.5% of the program memory used and that messages of types a, b and c above, which constitute more than 99% of the M-S communication, could be broadcasted concurrently with master-user communication, the disadvantages of using the serial register interface network outlined in parts 1 and 3 of Table II are reduced making a register interface superior to the DMA approach.

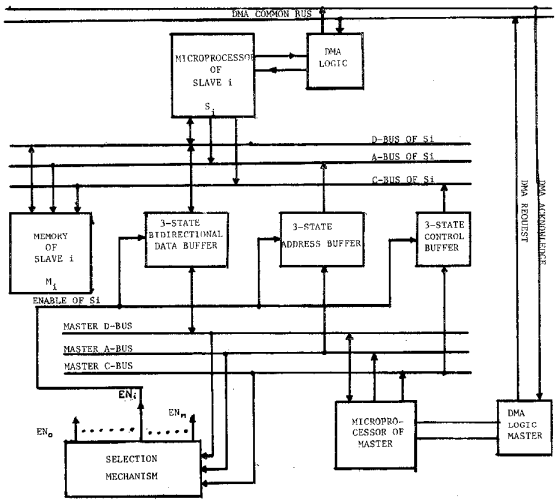


FIGURE 7 - DMA Interface for Master-Slave Communication

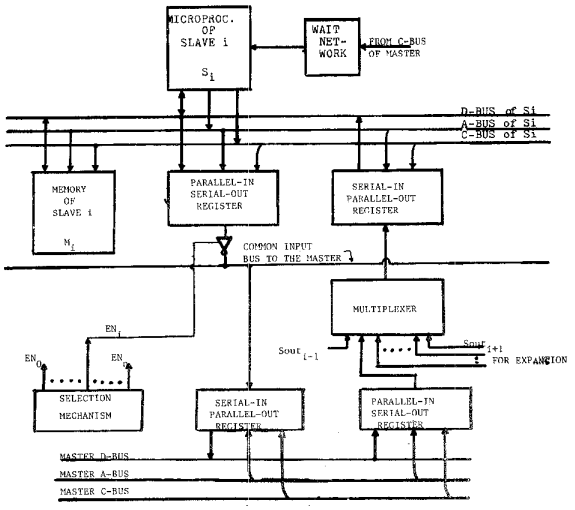


FIGURE 8 - Block Diagram of one Slave Corresponding to a Register Interface M-S Communication

TABLE II
Main Characteristics of DMA and Register Interfaces for a M-S Interconnection

	DMA		REGISTER
1.	Slaves play a passive role in the exchange, i.e. no receiving routine is required in the slaves	1.	Slaves play an active role in the transfer, so that the system requires more program memory than its DMA counterpart.
2.	A parallel link is implicit. As a consequence it results in a more expensive structure as shown in Table III.	2.	The choice between a parallel or a serial link is possible. A serial interface is much less costly than a DMA structure.
3.	Transmission is very fast, limited only by the flow rate of data between the memory of the master and the slaves memory.	3.	Transmission is slower. It depends on the bit rate and the length of the receiving routine in the slaves.
4.	M-S and S-S interfaces require different hardware, increasing the cost and decreasing efficiency.	4.	The corresponding hardware could be shared by S-S and M-S communications.

TABLE III
Comparison Between the Cost of a DMA and a Register Interface, as indicated by the Number of I.C.'s and Communication Wires

	DMA	REGISTER
# of Integrated Circuits	2(n+1) HEX Bidirectional 3-State Buffers	n/6 HEX Unidirectional 3-State Buffers
	3(n+1) HEX Unidirectional 3-State Buffers	1 8-Bit Serial-In Parallel Out Register
		1 8-Bit Parallel in Serial-Out Register
	5(n+1) I.C.'s	2+n/6 I.C.'s
# of Extra Wires Per Slave	16 Address Lines	1 Data In
	8 Data Lines	1 Data Out
	2 Control Lines	1 Enable
	3 DMA Lines	
	29 More Wires	3 More Wires

There are two possible implementations of the selection network of figure 8, either using a special hardware network in the master, as suggested in the figure, or with a software routine associated with the communication protocol. The latter approach is more flexible and economic but slower than the former due to the overhead required in identifying the type of message and specifying which slaves are to be contacted.

Again, if the broadcasting of most messages is done simultaneously with another activity of the master which precludes the operation of the slaves such that these two concurrent events do not interfere, then there will be no degradation due to the increase in overhead for the software approach. This being the case here, the decision was to use a software selection mechanism. Thus we will concentrate on defining a communication protocol and a message-processing routine in the slaves aimed at minimizing the overhead, the processing time for each message and the interference between M-S and master user communications. The resulting format for a message, is shown in figure 9.

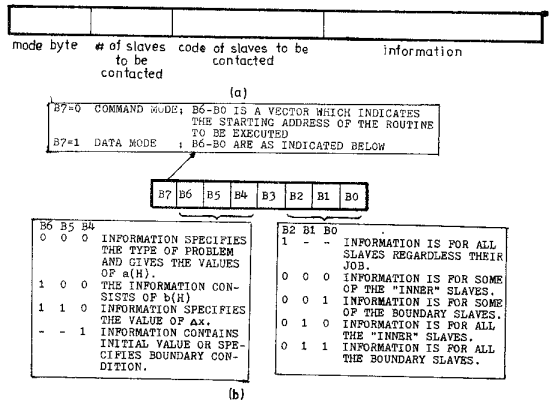


FIGURE 9 - Format of Messages in Master-Slave Communication (a) General Format of a Single Communication; (b) Detail of the 1st Byte Showing the Interpretation of Each Bit

EXPERIMENTAL RESULTS

A 1-dimensional prototype consisting of 9 slaves was constructed with capabilities for solving equation (1.1) for the following cases:

a) a and b are constants
b) a = a(H), b constant
c) a = a(H), b = b(H)

Each of these cases permits the following boundary conditions:

1. $H_N = f(t)$
2. H_N = constant
3. $\frac{\partial H_N}{\partial x} = f(H_N)$

Given the wide range of numbers involved in such problems, a floating-point data representation is mandatory. While the size of the exponential part is determined by the required range and must be of at least 6 bits, the size of the fractional part is given by the precision desired.

It order to estimate the degradation of precision due to the propagation of errors in the system, a simulation was conducted on a PDP-11/40 for different sizes of fractional part. The results of the simulation showed that the format finally chosen (1 byte of exponent in radix 2, two's complement notation and 2 bytes (16 bits) of fraction in two's complement notation normalized) produced results with an error of between 10^{-2} and 10^{-4}, depending on the type of problem, the nature of the boundary and the number of time steps.

It is believed that this format represents a good trade-off between practical accuracy and number of bits.

Among all the microprocessors readily available in our facility and reasonably suitable for the task, the National SC/MP-II was chosen. The SC/MP processor was found to yield a minimum-cost configuration as a result of its unconventional architectural features, but a configuration which would exhibit a relatively large solution time as a result of the rudimentary instruction set and limited internal communication paths of the processor. If the advantages of lowest cost and minimal packaging are ignored, the performance of a system build around almost most any other microprocessor, expressed as a function of its cost and speed, will be an improvement over that available with the SC/MP based system. However it was our feeling that a system offering the most extreme case of cost minimization brought two important advantages: both rapid implementation, and a dramatic measure of capabilities and limitations of a novel low cost organization.

Table IV shows the distribution of the cost of each P.E. among its component elements, as well as the number of packages and pins corresponding to each. Table V shows the average computational time per time step obtained with the prototype for the different cases listed above.

TABLE IV
Distribution of Cost Within Each Slave

COMPONENT	PRICE (%)	# OF I.C.'S	# OF PINS
CPU + CPU Logic	13.4	3	73
Program Memory	55	1	24
Data Memory	19	2	44
Communication Network	.1	1	16
Additional Logic	8.2	7	94
Power Supply	.6	1	3
Miscellaneous	3.7	-	-
	100%	15	253

TABLE V
Computation Time/Time Step for all Cases

Problem Types	Boundary Conditions: $H_N = F(t)$	$\frac{\partial H_N}{\partial x} = 0$	$\frac{\partial H_N}{\partial x} = f(H_N)$
a, b constant	34	34	35
a = a(H), b = constant	55	55	-
a = a(H), b = b(H)	70.7	70.7	71

T_c(MSEC)

A brief description of 3 of the tests performed to illustrate the precision that may be expected with this prototype as well as the trade-offs between speed and accuracy as a function of

the time and space increments, follows:

Test 1 - diffusion of a constant magnetic field through a plate of thickness 2a.

Figure 10-a compares the results obtained with a standard solution of the problem using a GPDC. Figures 10-b and 11 plot the results obtained by varying the time and space increments respectively. The overall solution time for this problem ranged between 4 sec. (for $\Delta x = 1/5$, $\Delta T = \Delta T$ max) and 34 sec. (for $\Delta x = 1/8$, $\Delta T = \Delta T$ max/10).

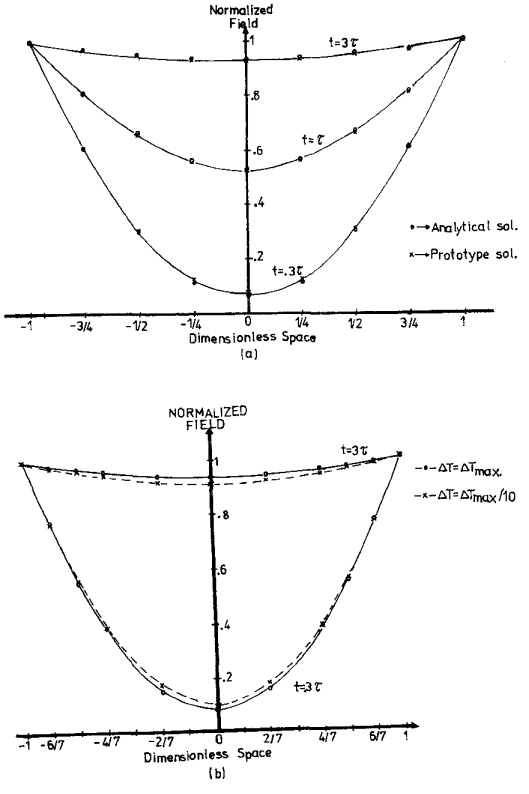


FIGURE 10 - Simulation of a Magnetic Diffusion Problem with Constant Permeability and Constant Input Boundary Conditions
(a) Comparison of the Results Obtained with the Prototype with an Analytical Solution of the Problem Calculated in a GPDC;
(b) Effect of Decreasing the Time Increment on the Accuracy of the Results

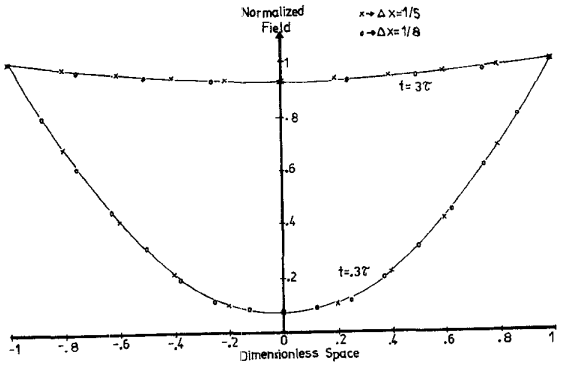


FIGURE 11 - Effect of Varying the Space Increment in Test 1

Test 2 - field distribution in a non-magnetic conductor. Equation (1.1) was solved with boundary conditions:

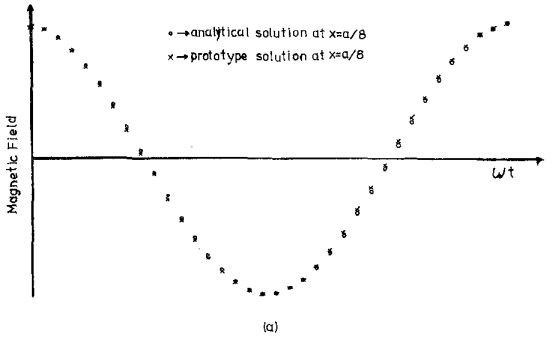
$$\frac{\partial H_o}{\partial t} = \omega \cos \omega t \qquad \frac{\partial H_N}{\partial x} = 0 \qquad (4.4)$$

Figure 12a shows the analytic solution at $x = a/8$, where a is the half width, and the results obtained with a 9 slave system. The error obtained ranged between 10^{-2} and 10^{-3} along the space dimension, as it did in the previous example. Figure 12b lists the error obtained in the peak value for different values of Δx while Figure 12c plots the time of solution against Δt expressed as a function of the ΔT maximum. One concludes that the improvement in accuracy that may be expected by decreasing ΔT does not justify the sacrifice in speed.

From Figure 12b one is tempted to use a large value of Δx since it produces faster results with equivalent accuracy than that provided with a smaller Δx. However, it must be noticed that Δx must be chosen so that a display of results at meaningful points in space is obtained. This choice is a function of the penetration depth (δ):

$$\delta = \frac{2}{abf} \qquad (4.5)$$

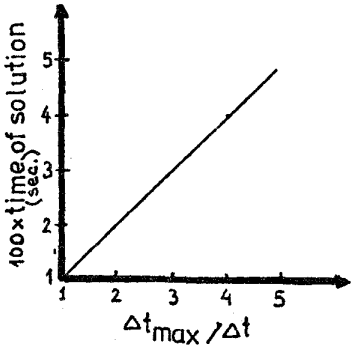
If a $\Delta x = \delta/10$ is considered adequate, a useful result could be provided in less than 90 sec. with $\Delta T = \Delta T$ max.



(a)

Δx	Solution Time	Absolute error	Relative error
a/5	81 sec.	.014	14 %
a/6	99 sec.	.009	9 %
a/7	114 sec.	.008	8 %
a/8	132 sec.	.004	4 %

(b)



(c)

FIGURE 12- Solution of a Magnetic Diffusion Problem with Constant Permeability and Time Varying Boundary Conditions
(a) Comparison of the Results Obtained with the Prototype at a Location 1/8 of the Way through the Sample Thickness a, with the Analytical Solution Evaluated with a GPDC at the Same Place;
(b) Relationship of Δx, Solution Time and Accuracy;
(c) Solution Time as a Function of ΔT

Test 3 - magnetic field diffusion in a magnetic conductor. The equations are the same as in test 2, but the permeability varies according to the expression:

$$\mu = \frac{B}{H} \text{ with } B = \frac{2B_S}{\pi} \, tg^{-1}\left[H/H_C\right] \qquad (4.6)$$

Figure 13 compares the result obtained with a 5 node simulation with the result obtained in a GPDC using an implicit solution of the problem. The differences in the peak value of the waveforms of figure 13 are less than .009 at any point in space. The computational time, however, was found to be very large, of the order of 214 sec.

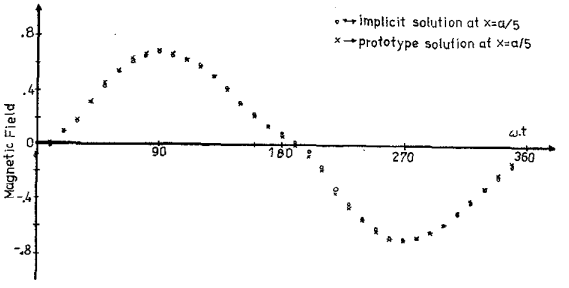


FIGURE 13 - Comparison Between the Results Obtained by a GPDC Working on an Implicit Solution and the Results Obtained with a 5-Slave Prototype for a Magnetic Diffusion Problem with Variable Permeability and a Time Varying Boundary

CONCLUSIONS

A special parallel processor has been presented for the solution of diffusion problems.

The choice of an explicit algorithm, the use of a master-slave organization with each slave mapping one point in the physical space, and the assignment of a private memory to each slave, ensures optimum utilization of resources with maximum modularity and simplicity, in comparison to solutions which use an implicit algorithm or common memory. Further, the added cost of modularity and simplicity will become insignificant in view of the rapidly reducing prices of semiconductor devices, particularly memory.

The choice of algorithm, organization of resources, and memory distribution, all together with the choice of communication networks and corresponding protocols, yield a modular structure that can be expanded to solve 2-dimensional problems and problems of diffusion in more than one variable with no penalty in execution time, merely by the simple addition of more slaves to the system bus.

Though, as a result of the arithmetic format chosen, the prototype operates with only 4 significant decimal digits, the final results obtained are equivalent for all practical purposes to those available with an IBM/370.

The modest solution speed of the system results directly from the choice of microprocessor made. However the speed of the initial prototype, while slow, is enough for many problems. It is very suitable for rapid solution of problems in which a and b of equation (1.1) are constant,

being for example always 20 times faster than real time for temperature problems. Upon reflection it is estimated that the use of a faster microprocessor would improve the speed by a factor between 8 (with a 6800 or 8080) and 20 (with the fastest version of a Z-80) at a corresponding cost increase of only 20% or so.

REFERENCES

(1) G.E. Forsythe, Finite Differential Methods for Partial Differential Equation, New York, Wiley 1967.

(2) B. Carnahan, Applied Numerical Analysis, New York, Wiley 1967.

(3) K. Oberretl, "Magnetic Fields, Eddy Currents and Losses Taking Variable Permeability into Account", IEEE Trans. Power Apparatus and Systems, Vol. 88, pp. 1646-1654., Nov. 1969.

(4) P. Silvester, "Network Analog Solution of Skin and Proximity Effect Problems", IEEE Trans. Power Apparatus and Systems, Vol. 86, pp. 241-247, Feb. 1967.

(5) K.K. Lim and P. Hammonds, "Flux Distribution in Saturated Steel Plates", Proc. IEEE, Vol. 119, pp. 1667-1674.

(6) E.A. Erdelyi, S.V. Ahmed and R.D. Burtness, "Flux Distribution in Saturated D-C Machines at No Load", IEEE Trans. Power Apparatus and Systems, Vol. 84, pp. 375-381, Oct. 1970.

(7) V.H. Kohne and G. Woelk, "The Digital Beuken Model-A Method for Determining Non-Stationary Heat Conduction Processes", Elektrowarme International, Vol. 27, pp. 302-308, Jul. 1969.

(8) J. Fong and C. Pottle, "Parallel Processing of Power System Analysis Problems Via Simple Parallel Microcomputer Structures", IEEE Trans. Power Apparatus and Systems, Vol. 97, Sept./Oct. 1978.

(9) W.L. Hatcher, F.M. Brasch Jr. and J.E. Van Ness, "A Feasibility Study for the Solution of Transient Stability Problems by Multiprocessor Strctures", Proc. IEEE Power Engineer Society Winter Meeting, New York, 1977.