A Binary Tree Multiprocessor: CORAL

YOSHIZO TAKAHASHI,* NAOKI WAKABAYASHI*
and YOSHIHIRO NOBUTOMO*

A feasible architecture for a highly parallel processing system which consists of 100 or more processors is studied. The commonly used shared data system in which a shared store is connected directly to the multiple of processors is inapplicable when the number of the processors is very large. An alternative system is the distributed data system in which the processors are loosely connected and the data are distributed to the processors prior to the processing. In this system the time to distribute the data and the time consumed in the inter-processor communications deteriorate the speed-up ratio. After various connections of the distributed data system, including star, chain, loop, lattice, and binary tree, were studied, it was revealed that the binary tree connection has the best performance. This binary tree multiprocessor is named CORAL. An algorithm to handle the inter-processor communications in the CORAL is developed. As possible applications of the CORAL, the parallel solutions of partial differential equations in one-dimensional heat conduction problem and the potential problem are studied. A prototype of CORAL consisting of seven microcomputers is described.

1. Introduction

There are two distinct methods in constructing a MIMD type parallel processing system. They are
1) the method in which the processors fetch data from a shared store when they are needed, and
2) the method in which data are distributed to the processors prior to the processing.

In a system constructed with the first method, which we call a shared data system, the access contention at the shared store deteriorates the processing speed. In a system constructed with the latter method, which we call a distributed data system, the time to distribute data is added to the processing time despite of avoiding contention. In a highly parallel processing system which consists of 100 or more processors, the shared data system is unfeasible. To our knowledge, the largest number of the processors in existing shared data systems is 16 [1]. In Chapter 2 we derive the speed-up ratios, data broadcasting times, and the average path lengths of various distributed data systems, and conclude that the binary tree distributed data system has a comparable speed-up ratio to the shared data system and has many favorable characteristics. The binary tree multiprocessor is named CORAL and its architecture is defined in Chapter 3. In Chapter 4 the solution of the partial differential equations in heat conduction and potential problems by CORAL is investigated. Chapter 5 describes the prototype of CORAL presently being developed at the Tokushima University.

2. Shared Data System Versus Distributed Data System

2.1 Shared Data System

In the shared data system, more than one working processors (WP) [1] are connected to a shared store as shown in Fig. 1. We denote the processing time of the system with n WPs by t_n, then

$$t_n = \frac{t_1}{n} + f(n). \tag{1}$$

That is, the processing time with n processors is one nth of that with one processor plus the overhead time $f(n)$ caused by the access contention for the shared store. The speed-up ratio s is obtained by

$$s = \frac{t_1}{t_n} = \frac{n}{1 + \dfrac{nf(n)}{t_1}}. \tag{2}$$

To make it simpler, the D/D/1 model is assumed to evaluate $f(n)$. Let the number of data being processed be M, the time to access a data in the shared store be d, and the time to process a data be T. We also introduce n_0 which is

$$n_0 = \frac{T}{d}. \tag{3}$$

It is then concluded that:

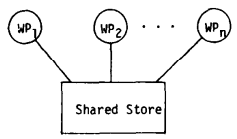

Fig. 1 Shared data system.

*Tokushima University, Tokushima, Japan.

(i) For $n \leqslant n_0$, the contention does not yet take place so that the waiting time to access a data in the shared store is d. As each one of the processors processes M/n data, t_n is obtained by

$$t_n = \frac{M}{n}(T+d) = \frac{t_1}{n}. \tag{4}$$

(ii) For $n > n_0$, the wait time increases to $(n-n_0+1)d$ due to the contention. t_n and s are obtained by

$$t_n = \frac{M}{n}(T+(n-n_0)d) = \frac{n+1}{n} Md. \tag{5}$$

$$s = \frac{T+d}{d}\frac{n}{n+1} = \frac{n_0+1}{n+1} n. \tag{6}$$

The speed-up ratio versus the number of processors is shown in Fig. 2. The upper bound of the speed-up ratio is (n_0+1).

As shown in Fig. 2, the increase of the number of the processors beyond n_0 does not much improve the speed-up ratio.

2.2 Distributed Data System

In the distributed data system, data are distributed to all working processors by the control processor (CP) prior to the processing. The time to distribute data depends on the connection of the processors in the system. Although certain time is required to deliver the calculated results back to CP, it is taken into account in the data distribution time. We will study the data distribution times and the speed-up ratios of the distributed data systems of different connections.

(i) Star Connection (Fig. 3)

In star connection, CP passes data to WP_1, WP_2, . . . , WP_n successively. Let the time to pass a data to WP be d, then the time until all WPs receive their data becomes $n(M/n)d$. The processing time of the system with n WPs is the time which the last WP receives the data and processes them. That is;

$$t_n = \frac{M}{n} T + Md. \tag{7}$$

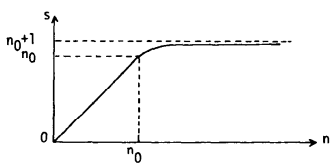

Fig. 2 Speed-up ratio of shared data system.

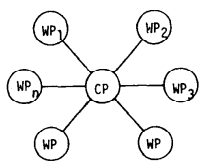

Fig. 3 Star connection.

The speed-up ratio is then,

$$s = \frac{M(T+d)}{M\left(\frac{T}{n}+d\right)} = \frac{n_0+1}{n_0+n} n. \tag{8}$$

Note that s is limited by (n_0+1) which is identical to the shared data system, although the saturation takes place much slower. This is observed in Fig. 9 which follows.

(ii) Chain Connection (Fig. 4)

In chain connection, CP sends data to WP_1 which passes them to the next WP except those addressed to itself. The next WP passes the received data to the next one until the last WP receives the data. When CP sends data in an order of WP_n, WP_{n-1}, . . . , WP_2, WP_1, the time until all WPs receive data is the least, which is

$$\frac{(2n-1)}{n} Md \fallingdotseq 2Md.$$

t_n and s are then

$$t_n = M\left(\frac{T}{n}+2d\right) \tag{9}$$

$$s = \frac{n_0+1}{n_0+2n} n. \tag{10}$$

The upper bound of s is $(n_0+1)/2$, which is one half of that of the star connection.

(iii) Loop Connection (Fig. 5)

In loop connection, CP sends data in both directions alternatively. The time until all WPs receives data is $2(n/2)(M/n)d = Md$. t_n and s are the same to those of the star connection.

(iv) Lattice Connection (Fig. 6)

In lattice connection, WPs are connected in a matrix with m rows and k columns. CP, which is connected to a WP at one of the corners, distributes all data to this WP which passes them to the other WPs at the top of each

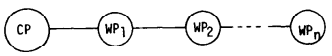

Fig. 4 Chain connection.

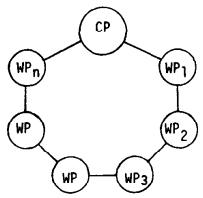

Fig. 5 Loop connection.

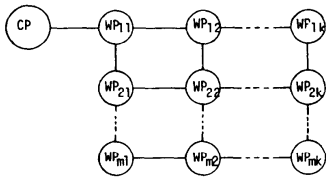

Fig. 6 Lattice connection.

column. These WPs then send data to the WPs belonging to their columns. The time until when all WPs receive their data is

$$\left((2k-1)\frac{M}{k}+2(m-1)\frac{M}{mk}\right)d \fallingdotseq 2M\left(1+\frac{1}{k}\right)d.$$

t_n and s are then

$$t_n=\frac{M}{n}T+2M\left(1+\frac{1}{k}\right)d, \tag{11}$$

$$s=\frac{M(T+d)}{\frac{M}{n}T+2M\left(1+\frac{1}{k}\right)d}=\frac{n(n_0+1)}{n_0+2n}. \tag{12}$$

s is bounded by $(n_0+1)/2$ which is the same to the chain connection.

(v) Binary Tree Connection (Fig. 7)

In binary tree connection, CP sends data in a sequence so arranged that all WPs are always ready to receive data when they are sent. In the connection of Fig. 7, this sequence may be 3–5–4–6–1–2 as described in Fig. 8. The time when all WPs receive data is $n(M/n)d=Md$ which is the same as that of the star connection. t_n and s are also the same as the star connection.

In Fig. 9, the speed-up ratios of various connections are plotted against the number of the working processors.

2.3 Broadcasting Time of Data

In the previous discussions, the data are assumed different for individual WPs. It is, however, probable that the WPs use some common data. The common data

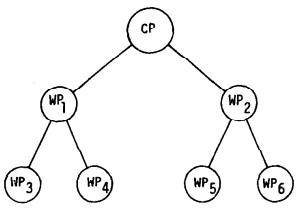

Fig. 7 Binary tree connection.

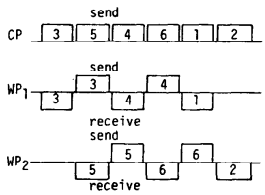

Fig. 8 Time chart.

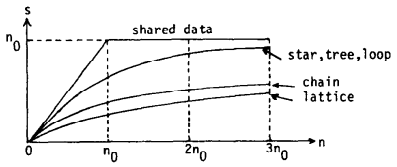

Fig. 9 Speed-up ratio comparison.

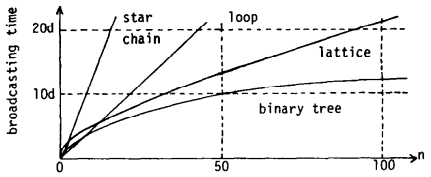

Fig. 10 Broadcasting time comparison.

are broadcasted from CP to WP by routing. Therefore the distribution time of the common data is much less than that of the individual data. The broadcasting times of the common data in various connections are obtained as follow.

(i) Star nd (13)

(ii) Chain nd (14)

(iii) Loop $nd/2$ (15)

(iv) Lattice $(m+k)d$ (16)

(v) Binary Tree $2(\log_2(n+2)-1)d$ (17)

These are plotted against n in Fig. 10 which reveals that the binary tree connection has the least broadcasting time when n is large.

2.4 Average Path Length

In a distributed data system the interprocessor communication is performed by exchanging messages between processors. The average distance between two processors of the system affects the efficiency of the interprocessor communication and the speed-up ratio in the consequence. The average path length is defined as the summation of the length of paths between all pairs of the processors divided by the number of all pairs of the processors. The average path lengths of various connections are calculated as follow.

(i) Star:

$$\frac{2n}{2}=2 \tag{18}$$

(ii) Chain:

$$\frac{\sum_{j=1}^{n-1} j(j+1)}{n(n-1)}=\frac{n+1}{3} \tag{19}$$

(iii) Loop:

$$\frac{2}{n-1}\sum_{j=1}^{\frac{n-1}{2}} j=\frac{n+1}{4} \quad (n\text{: odd}) \tag{20}$$

(iv) Lattice:

$$\frac{1}{km(km-1)}\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{h=1}^{m}\sum_{g=1}^{m}(|i-j|+|g-h|)=\frac{k+m}{3} \tag{21}$$

(v) Binary Tree:

$$\frac{1}{(2^{L+1}-1)(2^L-1)}\sum_{i=1}^{L}\left(2^i(2^{L+1-i}-1)\sum_{j=1}^{i}2^{L+1-j}\right)=\frac{(L-2)2^{2L+2}+(L+4)2^{L+1}}{(2^{L+1}-1)(2^L-1)} \tag{22}$$

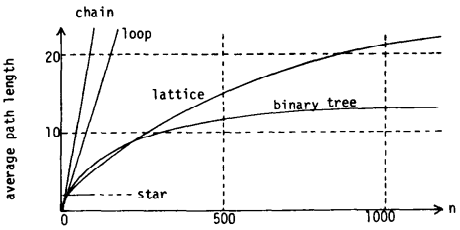

Fig. 11 Average path length comparison.

where L is the number of levels of a balanced tree. In Fig. 11 the average path lengths of the various connections are shown. The binary tree connection indicates an excellent performance when n is large.

2.5 Parallel Processings in Distributed Data Systems

The distributed data system consists of a CP and a set of WPs. The CP distributes data and program to WPs and collects the result of computation. The distributed data and programs are confined in the packets called job packet which we denote by

jobp[address, program, data]

where *address* is the identifier of the WP which processes *program* with *data*. The obtained result is returned to the CP in a form of an answer packet as denoted by

ansp[address, result].

The *address* is always that of the CP. Other packets called data packets are transmitted between WPs which we denote by

datap[address, data]

where *data* may include interprocessor messages.

The roles of the CP are, to split a job into job packets, to distribute them to WPs, and to collect the answer packets. Data packets are transmitted between WPs in the course of the processing. In the binary tree connection, sending job packets downward and answer packets upward are usual.

Although the general method of decomposing a job into a set of job packets is not yet established, the data-flow graph is helpful, through which we can reveal the data which immediately derive the final result, and also the data which derive the intermediate data until the input data are arrived. The nodes of the data-flow graph denote the data and the edges denote the processes. For instance, the calculation of $\sum_{i=1}^{1000} a_i b_i$ from the input data a_i, b_i $(i=1, 2, \ldots, 1000)$ is decomposed in a data-flow graph of Fig. 12. The job packet addressed to WP_i is

jobp [i, sum(j, k), ($a_j, \cdots, a_k, b_j, \cdots, b_k$)]

where

$k=100i$, and $j=k-99$.

Another example is the matrix multiplication problem of

$$X=A.B$$

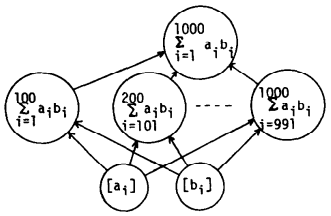

Fig. 12 Data-flow graph of $\sum_{i=1}^{1000} a_i b_i$.

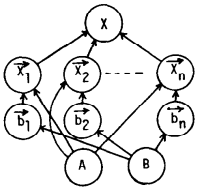

Fig. 13 Data-flow graph of matrix multiplication.

or

$$(\vec{x}_1\ \vec{x}_2 \cdots \vec{x}_n) = A \cdot (\vec{b}_1\ \vec{b}_2 \cdots \vec{b}_n). \tag{23}$$

The data-flow graph of this problem is shown in Fig. 13. The job packet for WP_i is

jobp [i, mult(i), (A, $\vec{b}_i$)]

where mult(i) is the program to calculate $A \cdot \vec{b}_i$ and return $\vec{x}_i$.

3. The Binary Tree Architecture

3.1 CORAL System

The binary tree architecture for highly parallel processing systems has many advantages. Among them are:

(1) The structure is recursive and constructing a highly parallel processing system including 100 or more processors is feasible.

(2) The structure of the element processor is simple, as it needes to provide only three ports for connection.

(3) The broadcasting time of the common data is short.

(4) The distribution time of the data is also short.

(5) The average path length is short especially when the parallelism is very large.

(6) No shared store is necessary.

After the previous discussions, we arrived at the conclusion that the binary tree is one of the most promising architecture for implementing a highly parallel processing system. The binary tree-structured parallel processor of Fig. 14 is named CORAL.

The processor at the root of the tree of CORAL is named a root processor (RP), the one at the node a node processor (NP), and the one at the leaf a leaf processor (LP). The tree may be either balanced or unbalanced. Normally RP serves as CP and NP and LP serve as WP. It is, however, admitted that one of NPs or LPs serves as CP whenever it is favorable (Fig. 15).

The number of the processors of a balanced CORAL of

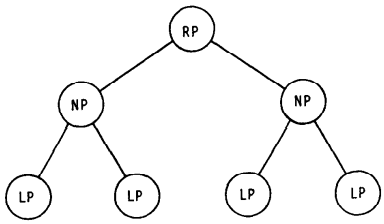

Fig. 14 CORAL system.

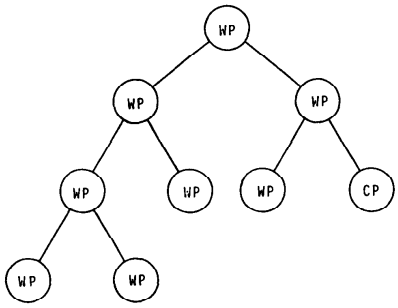

Fig. 15 Unbalanced CORAL system with CP at a leaf.

level L is given by

$$2^{L+1}-1 \tag{24}$$

This is shown in Table 1 for different levels. A parallel processing system of 1000 processors is obtained by CORAL with only 9 levels.

The NP has three paths which are top, left and right paths as shown in Fig. 16. They are denoted by T, L, and R respectively. RP has only L and R paths and LP has only T path. The processors of CORAL are given individual addresses. The address of RP is 1 and those of

Table 1 Number of processors of CORAL of different levels.

Level	Number of processor
0	1
1	3
2	7
3	15
4	31
5	63
6	127
7	255
8	511
9	1023
10	2047

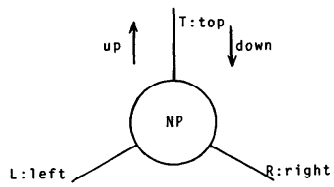

Fig. 16 Three paths of node processor.

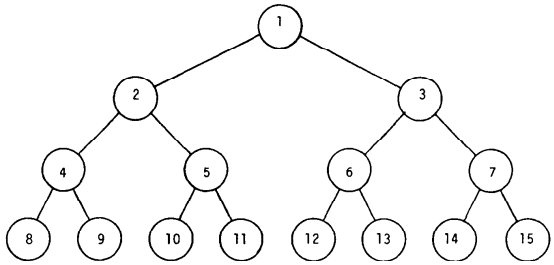

Fig. 17 Processor addressing.

other processors are as indicated in Fig. 17.

3.2 Interprocessor Communications in CORAL

The interprocessor communications in CORAL are performed by exchanging the data packets between processors. The packet has the destination address which is referred by RP and NP for the routing of the packets. The processor which received a packet, checks the destination address of the packet. If it matches its own address, the packet is received. If it does not, the destination address is divided by 2 and the fraction, if any, is dropped until the resultant address is either less than its own address, is equal to the address of the left successor which is twice of its own address, or is equal to the address of the right succesor which is twice of its own address plus 1. In the first case, the packet is sent to path T, in the second case to path L, and in the last case to path R. The algorithm of the routing of the packets is described in a McCarthy's conditional expression as follows.

$$\begin{aligned}\text{path}(a)=[&a=s\rightarrow S;\ a<s\rightarrow T;\ a=2s\rightarrow L;\\ &a=2s+1\rightarrow R;\ t\rightarrow\text{path}(a/2)]\end{aligned} \tag{25}$$

where a is the destination address, s is the address of the processor, and S is the path to the processor itself.

4. Applications of CORAL

4.1 Parallel Solution of One-Dimensional Heat Conduction Problem

The one-dimensional heat conduction problem is described in the following partial differential equation.

$$\frac{\partial U(x,t)}{\partial t}=c^2\frac{\partial^2 U(x,t)}{\partial x^2}\quad(0\leqslant x\leqslant a) \tag{26}$$

$$U(x,0)=f(x),\quad U(0,t)=f(0),\quad U(a,t)=f(a)$$

By expansion and denoting

$$r=\frac{kc^2}{h^2},\quad \text{where } h=\frac{a}{M} \tag{27}$$

and

$$U_n(m)=U(mh,nk), \tag{28}$$

the following difference equation is obtained.

$$U_{n+1}(m)=r\left[U_n(m-1)+\left(\frac{1}{r}-2\right)U_n(m)+U_n(m+1)\right]$$
$$(m=1, 2, \cdots, M) \quad (29)$$

When p processors are available, each processor evaluates s $(=M/p)$ equations of eq. (29). As the processors computing the adjoining regions have to exchange the boundary values at each interation, the strategy that those processors be located as near as possible is necessary.

One strategy of allocating the regions to the processors is thus. The tree is first expanded into a string by a tree walk as shown in Fig. 18 which contains 9 expanded nodes. The x-axis is divided into 9 regions and each of them is assigned to an expanded node as shown in Fig. 18. In this method, three regions are assigned to most processors, while only one region is assigned to a leaf processors. To balance the load of the processors, the size of the regions may be so adjusted that the total size of the regions assigned to each processor is the same.

Another strategy of allocation is as follows. We restrict that only one region is assigned to a processor and the path length between two processors, to which the adjoining regions are assigned, is kept as small as possible. We first divide the x-axis into as many equal regions as the number of the processors and each region is assigned to an individual processor. The region assigned to RP is so determined that the number of regions on its left side is equal to that of the processors connected under the left edge of RP. The number of the regions on the right side is also equals to that of the processors connected under the right edge of RP. These regions are assigned to the processors on the same sides. The rightmost region on the left side is assigned to the NP which is directly connected to the left edge of RP. The remaining regions are divided into left and right groups again and the left one is assigned to the processors connected under the left edge of the previous NP. In Fig. 19 the result of allocation for CORAL of level 3 is illustrated. It is revealed that most of the distance between the processors which compute the adjoining regions are 2.

The operation of CORAL is as follows. All processors are WP. When a processor receives from its predecessor the first and the last mesh points of the regions which are allocated the processors belonging to this branch, it

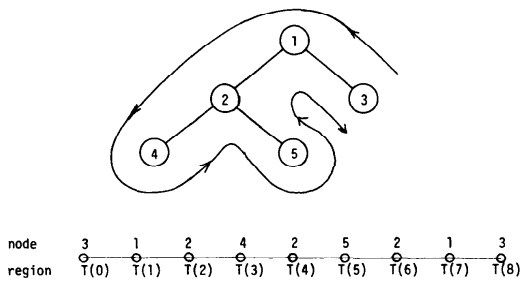

Fig. 18 Tree expansion.

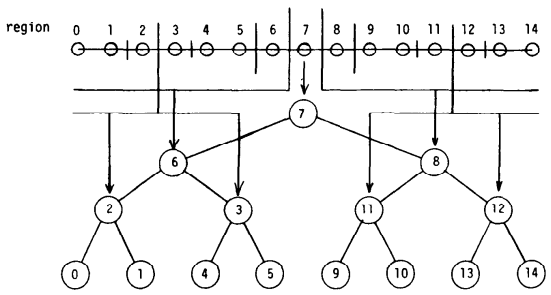

Fig. 19 Results of allocation.

divides the regions into three parts one of which is allocated to itself and the remaining parts are allocated to the successors. The processor sends the first and the last mesh points of the regions which are allocated to the processors under its left and right edges to the successors. It then calculates the initial values at each mesh point allocated to it. As a processor proceeds to compute $U_n(m)$ is requires that the boundary values be exchanged with the processors which are allocated with the adjoining regions. After n arrives at a predetermined value, the computed results are sent to RP. The NPs receive the results which are sent from the lower processors and forward them to RP. The operation of a NP is illustrated in the state transition diagram of Fig. 20.

The performance of CORAL is estimated by means of the computer simulation by obtaining the execution time of the solution of one dimensional heat conduction problem. The allocation strategy used in the simulation is the latter one. The simulated CORAL is the binary tree unbalanced only at the highest level. RP is used as CP. The assumed execution time of the element processor is estimated from that of 8080A.

The results of simulation are shown in Fig. 21 and in Fig. 22. In Fig. 21, the changes in the number of active processors versus time are indicated. In the begining zone where the number of active processors increases linearly, the allocated regions are broadcasted from RP to NPs. In the middle zone, the processors compute $U_n(m)$ and exchange boundary values at each time step. In the last zone, the calculated results are sent back to RP and NPs perform routing. In Fig. 22, the speed-up ratio versus the number of processors is indicated. Although saturation takes place when the number of processors exceeds 200, speed-up ratio of almost 300 is obtained when more than 600 processors are used.

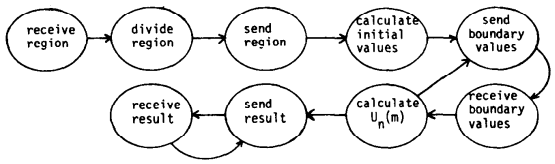

Fig. 20 State transition diagram of a node processor.

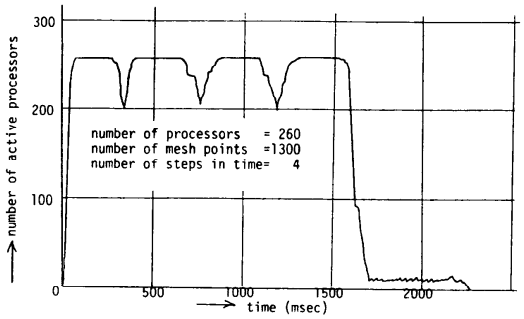

Fig. 21 Number of active processors versus time.

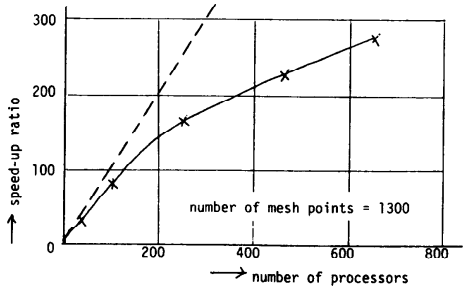

Fig. 22 Speed-up ratio versus number of processors.

4.2 Parallel Solution of Laplace Equation

The numerical solution of the Laplace equation requires iteration unlike the parabolic or hyperbolic partial differential equation. We will study the parallel solution of the following Laplace equation of two dimensions.

$$\nabla^2 U=\frac{\partial^2 U}{\partial^2 x}+\frac{\partial^2 U}{\partial^2 y}=0 \tag{30}$$

The approximate difference equation is

$$HU=h^2B \tag{31}$$

where H is a tridiagonal matrix and B is a boundary value vector.

Eq. (31) can be solved by the following iterative equations.

$$U_{ij}^{(k+1)}=\frac{a}{4}[U_{i-1,j}^{(k+1)}+U_{i,j-1}^{(k+1)}+U_{i+1,j}^{(k)}+U_{i,j+1}^{(k)}+h^2 b_{ij}] +(1-a)U_{i,j}^{(k)} \tag{32}$$

For the geometry of Fig. 23, eq. (32) becomes

$$U_1^{(k+1)}=\frac{a}{4}[U_2^{(k)}+U_4^{(k)}+h^2(b_1+b_{12})]+(1-a)U_1^{(k)}$$

$$U_2^{(k+1)}=\frac{a}{4}[U_1^{(k+1)}+U_3^{(k)}+U_5^{(k)}+h^2 b_2]+(1-a)U_2^{(k)}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$U_9^{(k+1)}=\frac{a}{4}[U_6^{(k+1)}+U_8^{(k+1)}+h^2(b_6+b_7)]+(1-a)U_9^{(k)} \tag{33}$$

The data-flow graph of eq. (33) is shown in Fig. 24. As

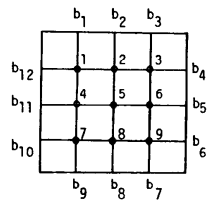

Fig. 23 Boundary and mesh points.

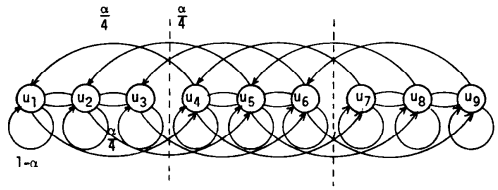

Fig. 24 Data-flow graph for Laplace equation.

the computation should be performed from left to right serially, it does not very well suit the parallel processing. To increase the parallelism of computation, we divide the entire region into three groups of (U_1, U_2, U_3), (U_4, U_5, U_6), and (U_7, U_8, U_9) and assign a processor to each one of the groups. Then these processors operate in pipeline. In other words, the mesh points are grouped by rows (or by columns) and each group is assigned to one processor. In this pipelining, the processors have to wait while others processing the adjoining groups of mesh points are operational, so that the two groups of processors operate in different phases. This constraint causes a degradation of the speed-up ratio to about one half. The convergence of the iterations may be detected when the last processor obtains the same result in a consecutive run.

5. CORAL Prototype

In order to prove the feasibility of the binary tree architecture in parallel processing, a CORAL prototype is being developed presently at the Tokushima University. The CORAL prototype is a two level binary tree multiprocessor consisting of 7 processors (Fig. 25). RP is

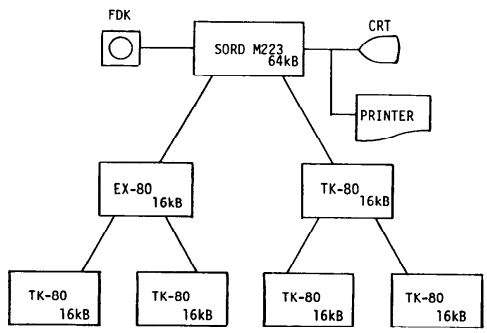

Fig. 25 CORAL prototype.

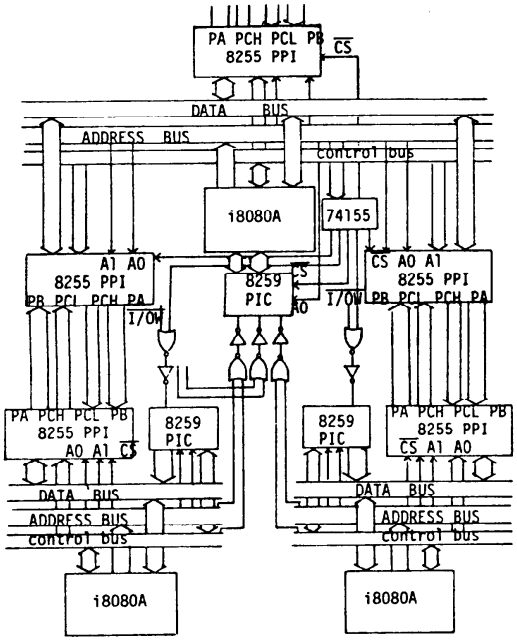

Fig. 26 Interprocessor connection of CORAL prototype.

SORD M223 Mark II microcomputer, and other processors are single board microcomputers.

The interprocessor connection is illustrated in Fig. 26, where three processors in connection are shown. The programmable peripheral interface 8255 and the programmable interrupt controller 8259 are used for the simplicity of the circuit.

6. Conclusion

The binary tree is a simple but a powerful structure. It has been observed in several computer architecture to date. Lipovski is probably the first one who has found the advantage of the binary tree structure in organizing a cpu [2]. A data-flow processor of Davis also imbeds a binary tree structure [3]. The tree organized multicomputer of Harris and Smith most resembles CORAL, although tertiary tree instead of binary tree is used. Among them, CORAL has the simplest structure and fully depends on the power of the binary tree structure unlike other ones.

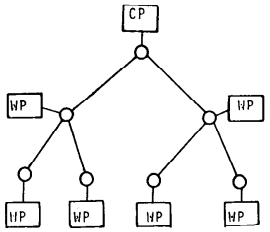

Fig. 27 Multiprocessor with a binary tree bus.

A further variation of the CORAL system is the one shown in Fig. 27 where the binary tree is used as a bus and processors are connected to the nodes of the binary tree bus. In this system, the routing of the packets is exclusively performed by the bus itself, so that the processors are relieved from this task.

The authors are grateful to the helpful suggestions made by Dr. Yoshimura of Toshiba Research and Development Center during the period of this research.

References

1. Takahashi, Y., Yoshimura, S. A Test Equipment for Parallel Program Processing, *Information Processing*, **20**, (1979) 319–322.
2. Lipovski, D. H. The Architecture of a Large Associative Processor, *Proc. SJCC* (1970) 385–396.
3. Davis, L. A. The Architecture and System Method of DDM1: A Recursively Structured Data Driven Machine, *Proc. fifth Annual Symposium on Computer Architecture*, (1978) 210–215.
4. Harris, J. A., Smith, D. R. Simulation Experiments of a Tree Organized Multicomputer, *Proc. Sixth Annual Symposium on Computer Architecture*, (1979) 83–89.

(Received June 12, 1980: revised August 6, 1980)