# 2 進木構造の並列処理システム CORAL

## A BINARY TREE MULTIPROCESSOR : CORAL

高橋義造　若林直樹　信友義弘

Yoshizo Takahashi, Naoki Wakabayashi, and Yoshihiro Nobutomo

徳島大学工学部

Tokushima University

### 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 the shared store when they are needed, and
2) the method in which data are distributed to the processors prior to processing.

In the first method, which we call the shared data method, the access contention at the shared store deteriorates the processing speed. In the latter method, which we call the distributed data method, the time to distribute the data is added to the processing time despite the contention is avoided.

In a highly parallel processing system which consists of 100 or more processors, the shared data method seems inapplicable. In Chapter 2 we investigate the speed-up ratios of various connections by the distributed data method and conclude that the binary tree architecture has the comparable speed-up ratio to that of the shared data system. The binary tree multiprocessor is named CORAL and its properties are studied in Chapter 3. In Chapter 4 some applications of the CORAL are discussed. Chapter 5 describes the prototype of CORAL presently being developed at the Tokushima University.

### 2. Shared Data vs. Distributed Data

#### 2.1 Shared Data

In the shared data system, more than one working processors WP [1] are connected to a shared store as shown in fig.1. This system is treated as M/D/1 problem of the queuing theory. We denote

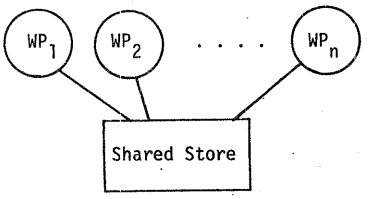


fig.1 Shared Data System

the processing time of the system with n working processors by $t_n$, then

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

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

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

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

$$n_0 = \frac{T}{\tau} . \qquad (3)$$

It is concluded that:

(i) For $n < n_0$, the contention does not occur and the wait time to access a datum in the shared store is $\tau$. As each one of the processors processes M/n data, $t_n$ is obtained by

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

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

$$t_n = \frac{M}{n}(T+(n-n_0)\tau) = \frac{n+1}{n} M\tau \qquad (5)$$

$$s = \frac{T+\tau}{\tau}\cdot\frac{n}{n+1} = \frac{n_0+1}{n+1}\cdot n. \qquad (6)$$

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

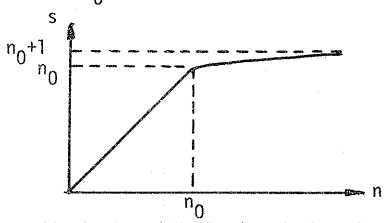


fig.2 Speed-Up Ratio of Shared Data System

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

2.2 Distributed Data

In the distributed data method, data are distributed to all working processors by the control processor (CP) prior to the processing. The time to ditribute data changes depending on the different connections. Although the calculated results may be delivered back to CP, this time is taken into account in the data distribution time. We will study the speed-up ratios in different connections of distributed data system.

(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 datum to a WP be $\tau$, then the time until the last WP receives data is

$$n\frac{M}{n}\tau = M\tau. \qquad (7)$$

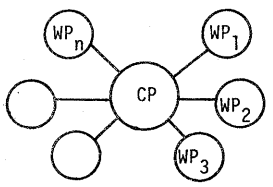


fig. 3 Star Connection

The processing time of the system of n WPs is the time which the last WP receives data and processes them. That is,

$$t_n = \frac{M}{n} T + M\tau. \qquad (8)$$

The speed-up ratio is then,

$$s = \frac{M(T+\tau)}{M(\frac{T}{n}+\tau)} = \frac{n_0+1}{n_0+n} n. \qquad (9)$$

Note that s is limited by $(n_0+1)$ which is identical to the shared data method, although the saturation takes place much slower. This will be found in fig.9.

(ii) Chain Connection (fig.4)

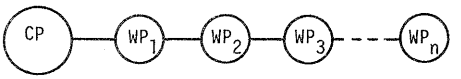


fig.4 Chain Connection

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

$$(2n-1)\frac{M}{n}\tau \simeq 2M\tau, \qquad (10)$$

so that $t_n$ and s are

$$t_n = M(\frac{T}{n} + 2\tau) \qquad (11)$$

$$s = \frac{(n_0+1)}{n_0+2n} n. \qquad (12)$$

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)

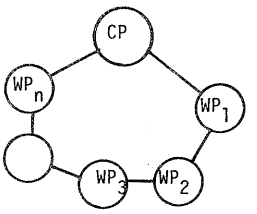


fig.5 Loop Connection

In loop connection, CP sends data to both directions alternatively. The time until when all WPs receive data is

$$2\frac{n}{2}\frac{M}{n}\tau = M\tau. \qquad (13)$$

$t_n$ and s are the same to those of the star connection.

(iv) Lattice Connection (fig.6)

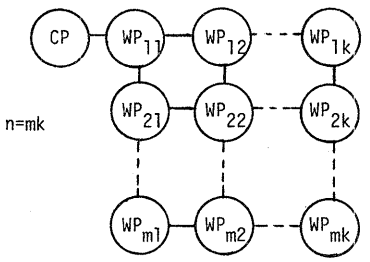


fig.6 Lattice Connection

In lattice connection, WPs are connected in a matrix of m rows and k columns. CP, connected to the WP at one of the corners, distributes all data to this WP which passes data to the WPs at the top of each column. These WPs then send data to the WPs belonging to their columns. The time until when all WPs receive data is

$$((2k-1)\frac{M}{k} + 2(m-1)\frac{M}{km})\tau \approx 2M(1+\frac{1}{k})\tau. \quad (14)$$

$t_n$ and s are then

$$t_n = \frac{M}{n}T + 2M(1+\frac{1}{k})\tau$$

$$s = \frac{M(T+\tau)}{\frac{M}{n}T + 2n(1+\frac{1}{k})\tau} \approx \frac{n(n_0+1)}{n_0+2n}. \quad (15)$$

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

(v) Binary Tree Connection (fig.7)

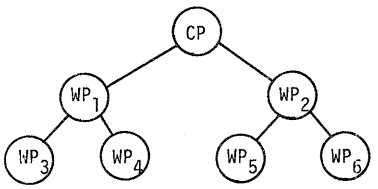


fig.7 Binary Tree Connection

In binary tree connection, the CP sends data in a sequence that all WPs are always ready to receive the data when they are sent. In the connection of fig.7, this sequence is 3,5,4,6,1,2 as described in fig.8. The time when all WPs receive data is

$$n\frac{M}{n}\tau = M\tau, \quad (16)$$

which is the same to that of the star connection.

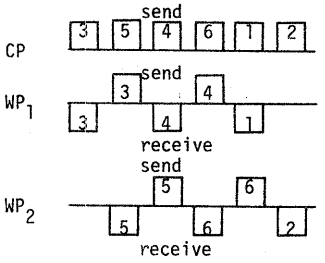


fig.8 Time Chart

In fig.9 the speed-up ratios of various connections are plotted versus the number of working processors.

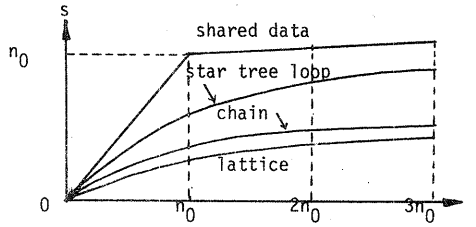


fig.9 Speed-Up Ratio Comaparison

2.3 Broadcasting of Common Data

In the previous discussion, the data are assumed different for different WPs. It is, however, probable that the WPs use common data. The common data are broadcasted from CP to WPs which, after storing them, pass to the next WPs. The distribution time of the common data is much less than that of the individual data.

The distribution times of the common data in various connections are obtained as follow.

(i) Star $n\tau$
(ii) Chain $n\tau$
(iii) Loop $n\tau/2$
(iv) Lattice $(k+m)\tau$
(v) Binary Tree $2(\log_2(n+2)-1)\tau$

The binary tree connection has the least common data distribution time . When n is large, the difference is remarkable.

2.4 Parallel Processing in Distributed Data Systems

The distributed data system consists of a CP and more than one WPs. The CP distributes data and program to WPs and collects the computed results. The distributed data and programs are confined in a packet called the job packet which we denote as

jobp[ address,program,data] (17)

where address is the identifier of WP which is designated to execute the program with the data. The obtained result is returned to CP if needed in a form of an answer packet,which is denoted as

ansp[ address,result] (18)

The address is always that of CP. Another packet which is tramsmitted between WPs is called a data packet which is

datap[ address,data] (19)

where data may include inter-processor messages.

The role of CP is to split the job into job packets,to distribute them to proper WPs ,and to collect the answer packets. Job packets and answer packets may be transmitted among WPs when needed. In the binary tree connection, sending job packets downward and answer packets upward seems efficient.

Althogh the general method of decomposing a job into job packets is not yet found, the data-flow qraph is useful, through which we find the data which immediately derive the final result, and then find data which also immediately derive the previously mentioned data until the input data are arrived. This is shown in fig.10.

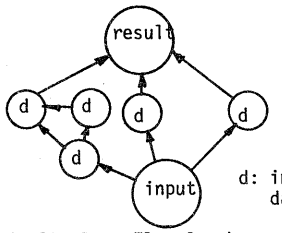


fig.10 Data-Flow Graph

In fig.10, nodes denote the data and arcs denote the process. For instance, the calculation of $\sum_{i=1}^{1000} a_i b_i$ from the input data $a_i, b_i$ (i=1,2,.., 1000) is decomposed in the data-flow graph of fig.11.

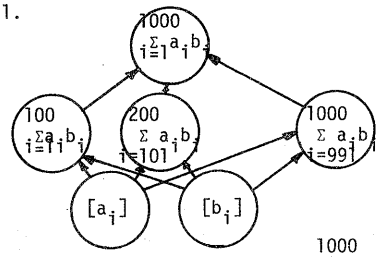


fig.11 Data-Flow Graph of $\sum_{i=1}^{1000} a_i b_i$

The job packet addressed to WPi is

jobp[i,sum(j,k),($a_j$,..,$a_k$,$b_j$,...,$b_k$)] (20)

where j=100i-99, k=100i.

Another example is the matrix multiplication problem of

$$X = A.B$$

or

$$(\vec{x}_1\ \vec{x}_2 \ldots \vec{x}_n) = A.(\vec{b}_1\ \vec{b}_2 \ldots \vec{b}_n) \quad (21)$$

The data-flow graph of this problem is shown in fig.12. The job packet for $WP_i$ will be

jobp[i,mult(i),(A,$\vec{b}_i$)] (22)

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

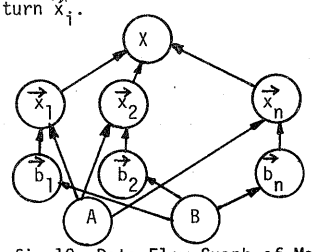


fig.12 Data-Flow Graph of Matrix Multiplication

## 3. Binary Tree Architecture and its Properties

### 3.1 CORAL System

The binary tree architecture for highly parallel processing systems has many advantages. They 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. It has only three ports for connection.

(3) The broadcasting time of the common data is shortest among other structures.

(4) The distribution time of the data is among the shortest.

(5) No shared store is necessary.

As a result of the prevoius discussions, we have concluded that the binary tree is one of the most promising architecture for implementing the highly parallel processing system. The binary tree-structured parallel processor of fig.13 is named CORAL.

The processor at the root of the tree is called the root processor (RP), and the processor at any node is called the nodal processor (NP).

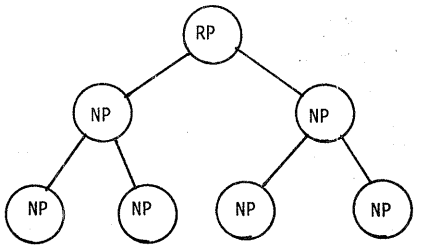


fig.13 CORAL System

The tree may be either balanced or unbalanced. Normally RP is used as CP and NPs are used as WPs. When it is necessary, however, a NP may be used as CP as shown in fig.14.

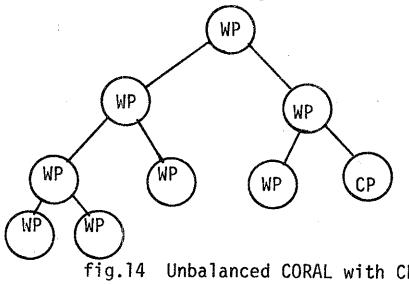


fig.14 Unbalanced CORAL with CP in node

In Table I, the number of the NPs of the balanced CORAL of different levels are shown.

Table I

| level | number of NP |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 6 |
| 3 | 14 |
| 4 | 30 |
| 5 | 62 |
| 6 | 126 |
| 7 | 254 |
| 8 | 510 |
| 9 | 1022 |

Let the numbers of NPs be $N_p$ and levels of tree be n, then

$$N_p = 2^{n+1} - 2 \qquad (23)$$

One NP has three paths which are top, left and right paths as shown in fig.15.

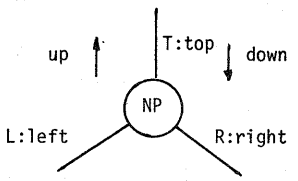


fig.15 Three Paths of Nodal Processor

The processors of CORAL have addresses. RP has address 0. The address of the NP connected to the left arc of RP is 20..0 and that of the NP connected to the right arc is 10..0. The number of digits is equal to the level of the tree. The address of other NP is determined by adding 2 to the last nonzero digit of the address of the preceding NP if it is connected to the left arc of that NP,and 1 if it is connected to the right arc. In fig.16 the addresses of the processors of level 3 CORAL are shown.

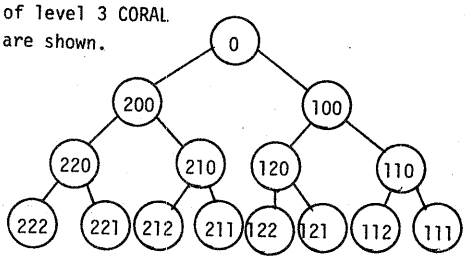


fig.16 Addresses

3.2 Average Path Length

In a distributed data system, the inter-processor communication is performed by exchanging messages between processors. The average processor distance of the particular connection effects the speed of the inter-processor communication of the system. The average path lengths of various connections are calculated as follow.

(i) Chain

$$\text{average path length} = \frac{\text{total path length}}{\text{number of paths}} = \frac{\sum_{j=1}^{n-1} j(j+1)}{n(n-1)} = \frac{n+1}{3} \qquad (24)$$

(ii) Loop

$$n+2n+3n+\ldots+\frac{n-1}{2}n = \frac{n+1}{4} \quad (n:\text{odd}) \qquad (25)$$

(iii) Star

$$2n/n = 2$$

(iv) Lattice

$$\frac{1}{km(km-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{h=1}^{m} \sum_{g=1}^{m} [|i-j| + |h-g|] = \frac{k+m}{3} \qquad (26)$$

(v) Binary Tree

$$\frac{2a_n}{n(n-1)} \qquad (27)$$

where $a_n = a_{n-1} + b_{n-1}$ , $a_2 = 1$

$b_n = b_{n-1} + c_{n-1}$, $b_1 = 3$

$c_n = 0$ for n=0

$= m+1$ for n>=2 and k=0

$= 2^{j+2}-2$ for n>=3 and k≠0

$n = m+k$

$m = 2^{N+1}-2$, where N is the level of tree

$k = 2^j(2i+1)$

In fig.17 the average path lengths of the various connections are shown. The binary tree connection indicates an excellent performance for large n.

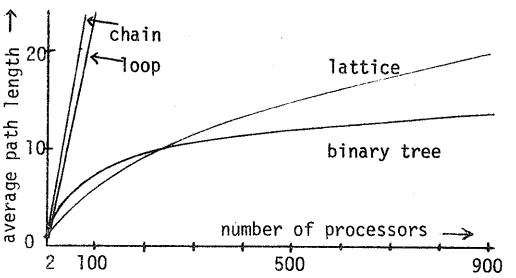

fig.17 Average Path Length Comparison

### 3.3 Interprocessor Communications in CORAL

The interprocessor communications in CORAL are executed by transmitting the data packets between processors. The packet has the designation address, and the routing of the packet is handled by RP and NP by checking this address. The RP checks the first digit of the address of the received packet and if it is 2 sends the packet to L direction, if it is 1 sends to R direction and if it is 0 receives it. The NP, as it receives a packet, compares the designation address with its own address. If they match the packet is received. If they unmatch,the packet is sent to T direction or thrown up. If the own address matches to the leading digits of the designation address, the following digit is checked. If it is 2 the packet is sent to L direction and if it is 1 it is sent to R direction. The routing algorithm of NP is shown below.

```
type address=array[0..9] of digit;
var  own,destination:address;
     path:(T,L,R);
     match:boolean;i:integer;
begin
   i:=0;
   match:=true;
   repeat
      if own[i] =designation[i]
         then i:=i+1;
         else match:=false;
   until own[i]=0 or match=false;
   if match=false
      then path:=T;
      else case destination[i] of
              0: accept packet;
              1: path:=R;
              2: path:=L;
           end;
end.
```

## 4. Application of CORAL

### 4.1 One Dimensional Heat Conduction Problem

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

$$\frac{\partial T(x,t)}{\partial t} = c^2 \frac{\partial^2 T(x,t)}{\partial x^2} \quad (0 \leq x \leq a) \qquad (28)$$

$T(x,0)=f(x)$

$T(0,t)=f(0)$, $T(a,t)=f(a)$

By expansion and let

$$r = \frac{ka^2}{h^2} = \frac{1}{6} \quad \text{where } h=a/M \qquad (29)$$

we obtain

$$T(x,t+k) = \frac{1}{6}[T(x+h,t)+4T(x,t)+T(x-h,t)] + O(h^6) \qquad (30)$$

Neglecting higher orders of h and denoting

$$T_n(m) = T(mh,nk) \qquad (31)$$

the following difference equation is obtained.

$$T_{n+1}(m) = \frac{1}{6}[T_n(m-1)+4T_n(m)+T_n(m+1)] \quad (m=1,2,3...M) \qquad (32)$$

When p processors are available, each processor computes s(=M/p) equations of (32). As the processors computing the adjoining T have to exchange the obtained boundary values at each iteration, these processors should be located as near as possible.

One method of allocating the region to NP is as follows. The tree is expanded into a string as shown in fig.18 which contains 9 expanded nodes. The x axis is divided into 9 regions and each region is assigned to an expanded node as shown in fig.18. In this method most

processors are assigned three regions, while the processor at the leaf is assigned only one region. To balance the load of the processors, three regions may be assigned to the processors at the leaf.

Another method of allocation is thus. We restrict that only one region is assigned to each processor and the distance of the processors computing the adjoining regions is kept as small as possible. We first divide the regions into

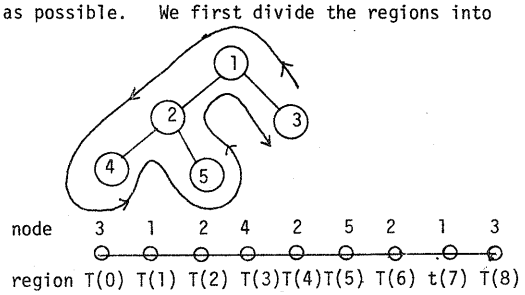


fig.18 Tree Expansion

the number of the processors and the processor at the root is assigned a particular region such that the number of regions to the left of that region is equal to the number of the processors connected to the left arc of the root processor. These regions which situate to the left of the region which is assigned to the root processor are assigned to the processors which are connected under the left arc of the root processor. The processor which is directly connected to the left arc of the root processor takes the rightmost region and the remaining regions are divided into two groups. The left group is assigned to the processors connected under the left arc of that processor. The regions which situate to the right of the particular region which is assigned to the root, are assigned to the processors connected to the right arc of the root. The processor which directly connects to the root takes the leftmost region and the remaining regions are divided into two groups and the left one is assigned to the processors which are connected to the left arc of this processor. In fig.19 the result of the allocation for the tree of level 3 is shown. It is revealed that the most of the distances between the processors which compute the adjoining regions are 2.

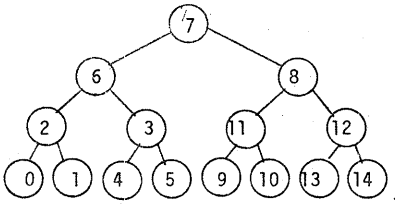


fig.19 Result of Allocation

4.2 Lapalace Equation

The numerical solution of the Lapalace equation requires iteration unlike the parabolic or hyperbolic partial differential equations. We will consider the following Lapalace equation in two dimensions.

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad (33)$$

The approximate difference equation is

$$H.\vec{u} = \vec{B} \qquad (34)$$

where H is a tridiagonal matrix and $\vec{B}$ is the boundary value vector.

Eq(34) is solved by the following iterative equations.

$$u_{ij}^{(k+1)} = \frac{\alpha}{4}[u_{i+1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i-1,j}^{(k)} + u_{i,j-1}^{(k)} + b_{ij}] + (1-\alpha)u_{ij}^{(k)} \qquad (35)$$

For the boundary of fig.20, eq(35) becomes

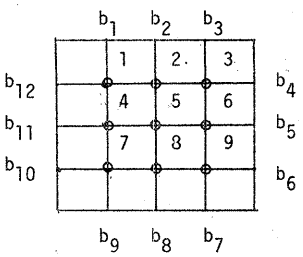


fig.20 Boundary and Mesh Points

$$\left.\begin{aligned} u_1^{(k+1)} &= \frac{\alpha}{4}[u_2^{(k)} + u_4^{(k)} + h^2(b_1 + b_{12})] + (1-\alpha)u_1^{(k)} \\ u_2^{(k+1)} &= \frac{\alpha}{4}[u_1^{(k+1)} + u_3^{(k)} + u_5^{(k)} + h^2 b_2] + (1-\alpha)u_2^{(k)} \\ &\cdots\cdots\cdots\cdots \\ u_9^{(k+1)} &= \frac{\alpha}{4}[u_6^{(k+1)} + u_8^{(k+1)} + h^2(b_6 + b_7)] + (1-\alpha)u_9^{(k)} \end{aligned}\right\} \qquad (36)$$

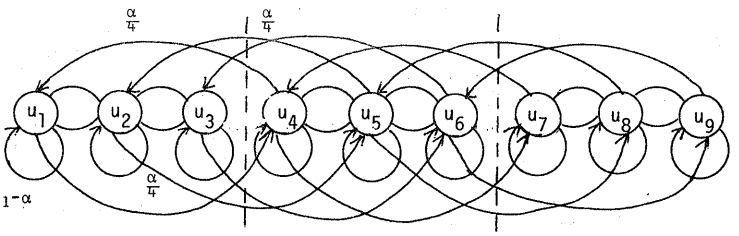


fig.21 Data-Flow Graph for Laplace Equation

The data-flow graph of eq(36) is shown in fig.21. As the computation has to be performed from left to right serially, it does not well suit to the parallel processing. Therefore we divide the whole region into 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 they can process in pipeline. This means that the mesh points are grouped by the rows (or columns) and each row is assigned to one processor. In this pipelining, the processors have to wait while the adjoining processors are operational. This causes degradation of the efficiency to one half. The convergence of the iteration will 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 multiprocessor, a CORAL prototype is being developed presently. The CORAL prototype is a 2 level binary tree consisting of one CP and 6 WPs as shown in fig.22. The CP is the SORD M223 Mark II microcomputer with 64 kB of memory,one of the WP is EX-80 one board computer and the other WPs are TK-80.

The interprocessor connection is described in fig.23 which shows three nodal processors in connection. The programmable peripheral interface 8255 and the programmable interrupt controller 8259 are used for simplicity.

## 6. Conclusion

The binary tree is a simple but powerful structure. It has been considered in several computer architecture to date. Lipovski is the first who found the advantage of the binary tree structure in organizing the cpu. [2] A data-flow machine of Davis also imbeds binary tree structure. [3] The tree organized multicomputer of Harris and Smith most resembles CORAL although tertiary tree instead of binary tree is used. [4] The CORAL, however, is the most simple structured system and fully relies on the power of binary tree structure unlike the other tree structured systems.

The authors are grateful to the suggestions of Dr. Susumu Yoshimura of Toshiba R & D Center for this study.

References:

1) 高橋・吉村「並列プログラム処理実験装置」情報処理 Vol. 20, No. 4, pp. 319- 322, 1979 年 4 月

2) Lipovski,D.H.,"The Architecture of a Large Associative Processor",Proc.SJCC,pp.385-396, 1970

3) Davis,L.A.,"The Architecture and System Method of DDM1: A Recursively Structured Data Driven Machine",Proc. Fifth Annual Symp. on Computer Architecture,pp.210-215,1978

4) Harris,J.A.,Smith,D.R.,"Simulation Experiments of a Tree Organized Mulicomputer",Proc. Sixth Annual Symp. on Computer,pp.83-89,1979

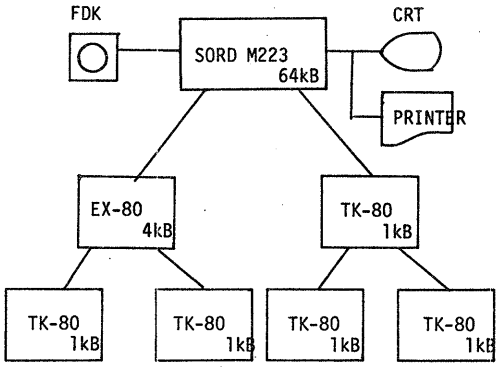


fig.22 CORAL Prototype

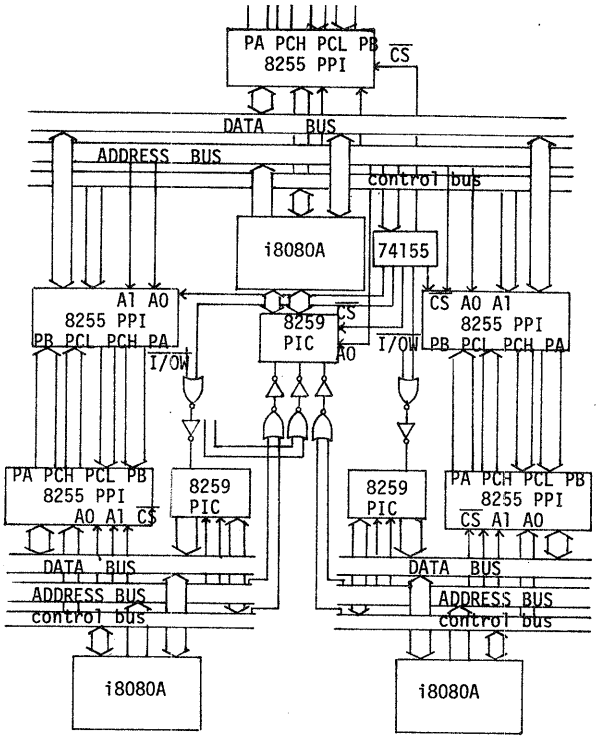


fig.23 Interprocessor Connection of CORAL