ADINA Computer II : I. Architecture and Theoretical estimates

AUTHOR(S):

NOGI, Tatsuo

CITATION:

NOGI, Tatsuo. ADINA Computer II : I. Architecture and Theoretical estimates. Memoirs of the Faculty of Engineering, Kyoto University 1981, 43(1): 124-144

ISSUE DATE:

1981-03-25

URL:

http://hdl.handle.net/2433/281168

ADINA Computer II*

I. Architecture and Theoretical estimates

By

Tatsuo Nogi**

(Received September 30, 1980)

Abstract

This paper gives a proposal about a new parallel computer of a MIMD array type, being suitable especially for high speed computing in engineering and science. It has a master unit, N submaster units and N^2 slave units, having their own processors and private memories respectively. Also, there are common memories between every submaster and every corresponding slave and further buffer memories used for transferring data between any allowed pair of slaves. In comparison with the ADINA Computer I [1] proposed previously, the new computer (ADINA Computer II) is useful for not only one and two dimensional problems, but also for three dimensional problems. Furthermore, it more easily allows an increase of the number of processors. Theoretical estimates for some examples of computation show that the ADINA Computer II is also very efficient.

1. Introduction

This paper is about a new parallel computer of a MIMD array type, suitable for high speed computing in engineering and science. It is a multi-processor system with several buffer memories provided for transferring data among slave units.

Up to this time, several kinds of multi-processor systems of array types have been developed especially for solving partial differential equations numerically. Among them, ILLIAC-IV is a representative machine. It is literally a multi-processor system of a two dimensional array type, with a network of data buses for providing a path only between every pair of adjacent processors. Hence it has an essential defect that it is troublesome and takes a long time to transfer data between any two separated processors.

Another mu'ti-processor system with a cross-bar switch permitting a direct transfer of data between any two processors does not suffer a loss of throughput in a data transfer. However, as it usually needs a vast sum of hardware, it has been

---

* Alternating Direction Immediate Nexus Array Computer II

** Department of Applied Mathematics and Physics

avoided for usage.

The most important problem how to exchange data among processors when it is intended to design a parallel computer. The author previously proposed a parallel computer (ADINA Computer I) with a two dimensional array of buffer memories, which makes it possible to exchange data between any two processors, but does not increase its quantity of hardware so much. (See [1]). Nevertheless, it is again a difficult problem to increase the number of processors in the same configuration.

The ADINA Computer II is intended to hold an advantage over the ADINA Computer I, and yet not to increase the hardware so much. On the ADINA Computer II, a direct exchange of data is not allowed for every pair of processors, but an indirect exchange is always possible if another processor is used as a mediator. Also, it is noticed that, if the number of slave processors is N^2, the number of buffer memory blocks in the ADINA Computer II is N^3 while that in the ADINA Computer I is N^4.

2. Architecture

The ADINA Computer II is a hierarchy composed of a master processor, N submaster processors and N^2 slave processors. The master processor makes a synchronization of the whole system and plays the role of an INPUT/OUTPUT processor. The submasters are connected to it directly through channels. To every submaster, corresponding N slaves are connected by each common memory which comes from both sides of the submaster and the slaves. Every submaster makes a synchronization among its slaves and plays the roles of a distributor of programs and a mediator of data between the master and its slaves.

The ADINA Computer II is provided with a fairly vast buffer memory. A capacity of memory is taken as a unit which is called a memory block. Another unit is a buffer memory board which is composed of N^2 memory blocks. The whole buffer memory consists of N boards with the number $k=1, 2, \cdots, N$. The k-th board is for bufferring when transferring data bi-directionaly (alternatively in a direction) between one slave on the k-th row and one on the k-th column of the two dimensional array of slaves. As a component of the buffer memory, either a RAM (Random Access Memory) or a FIFO (First-In-First-Out memory) may be used. In either case, a hardware structure may be suited for either bi-directional or uni-directional read/write ability. In view of efficiency, the RAM is superior to the FIFO and the bi-directional structure is superior to the uni-directional one. On the other hand, in view of the quantity of hardware, it is less for the FIFO

than for the RAM, and it is less for the uni-directional structure than for the bi-directional one. In the case of using the FIFO, a transfer speed is, however, not so slow in comparison with the case of the RAM, if a DMA channel is set between every slave and every FIFO.

The only case considered in this paper is that the RAM is used for the buffer memory, and it is accessed from both sides of the row and column positions of slaves. Then, it is used not only as the buffer memory but also as a memory for working.

Besides, all processors are pvrovided with private memories.

A method of bus connection embodied in the ADINA Computer II will be cleared by referring to some accompanying figures as follows: MU denotes the master processor, SU-$((k))$ (or simply $((k))$; $k=0, 1, \cdots, N-1$) denotes the k-th submaster processor and AU-$((j,k))$ (or simply $((j,k))$; $j,k=0, 1, \ldots, n-1$) denotes the slave master on the cross point of the k-th row and the j-th column in the two dimensional array. Further, $(i,j)_k$ (i,j, $k=0, 1, \cdots, N-1$) denotes the (i,j)-th memory block on the k-th memory board.

Fig. 1 shows an outline of the bus connection between MU and every SU, and

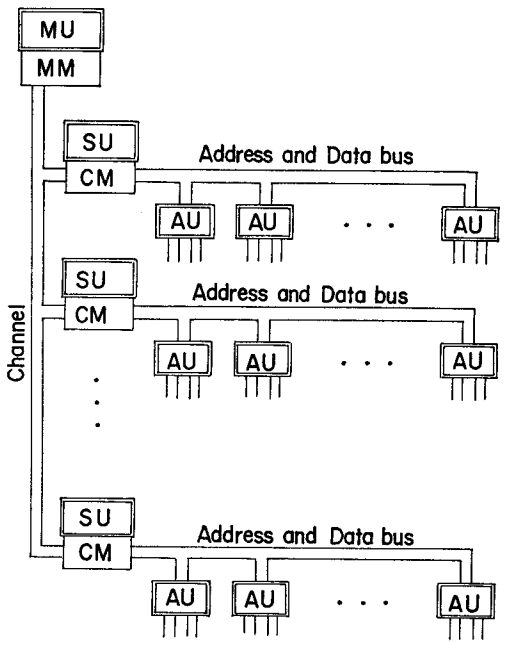

Fig. 1. Bus connection of MU-SU and SU-AU

between every SU and the corresponding AU's. Here MM is the main memory of MU, and CM is a common memory accessible from both sides of a SU and

the corresponding AU. An access by an AU is allowed under control of its master SU. Two other bauses from every AU, seen in Fig. 1, go to two corresponding memory blocks respectively. The details are seen in Fig. 2.

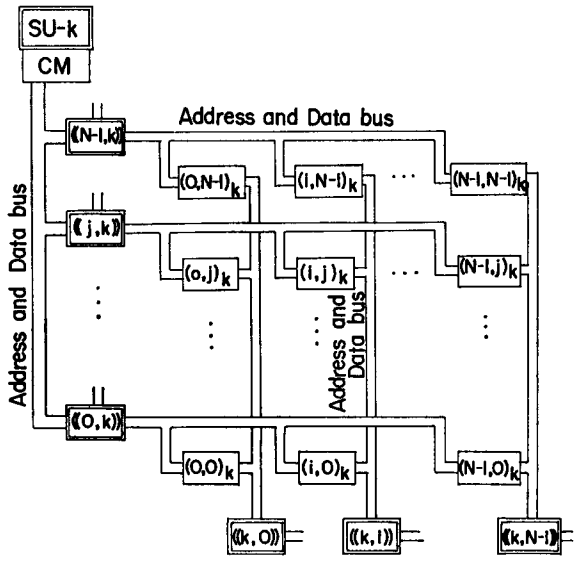

Fig. 2. Bus connection between AU and the buffer memory

Fig. 2 shows a subsystem composed of SU-$((k))$, AU-$((j,k))$ $(j=0,1,\cdots,N-1)$, AU-$((k,i))$ $(i=0,1,\cdots,N-1)$ and the k-th memory board. Each block $(i,j)_k$ is accessed only by two processors $((j,k))$ and $((k,i))$. Their competition of access is helped by the control of $((k))$.

Usually, the control is taken in the following way. At a time, every processor $((j,k))$ $(j=0,1,\cdots,N-1)$ accesses corresponding buffer memory blocks $(i,j)_k$ $(i=0,1,\cdots,N-1)$ together, and at another time every processor $((k,i))$, $(i=0,1,\cdots,N-1)$ accesses corresponding blocks $(i,j)_k$ $(j=0,1,\cdots,N-1)$ togeher. Besides, it is also possible for a AU to write data in a whole buffer memory row or column in a broadcasting manner.

Another bus from every AU, seen in Fig. 2, goes to another corresponding buffer memory board.

By such a manner of bus connection, a direct transfer of data through the k-th memory board is permissible only between one of $((j,k))$ $(j=0,1,\cdots,N-1)$ and one of $((k,i))$ $(i=0,1,\cdots,N-1)$. Such a manner of connection is described by the schema

$$((j,k))—(i,j)_k—((k,i))\,.$$

As a rule, it is expressed as follows:

i) A direct transfer of data through the k-th buffer memory board is possible between two processors with the number k in the brackets $((\ ,\ ))$ before and after their commas respectively.

ii) It is, then, the buffer memory $(i,j)_k$ that stands, for example, between two processors $((j,k))$ and $((k,i))$. The name is indicated in the following manner: to shift the pair of numbers in a processor's name in such a way so as to put out the number k common to those processors from each double bracket at once, to make the removed number k a suffix of a single bracket with the number left like that $(\ ,j)_k$ or $(i,\ )_k$ and to put the number i or j, being a counterpart to the common number k, in the double bracket of the partner processor into the blank of the single bracket.

It is, however, impossible to transfer data directly even between any two processors on the same row or column, for example, between $((j,k))$ and $((j-1,k))$.

Nevertheless, an indirect transfer is always allowed between any two AU's, only if another AU is used as a mediator. In fact, a transfer from $((j,k))$ to $((l,m))$ is done in the following way: $((j,k))$ writes the data in the buffer memory block $(l,j)_k$, then $((k,l))$ reads them and writes again in $(m,k)_l$, and finally $((l,m))$ reads them. The procedure is described in such a shema as

$$((j,k)) \rightarrow (l,j)_k \rightarrow ((k,l)) \rightarrow (m,k)_l \rightarrow ((l,m))\ .$$

Therefore it is found that such an indirect transfer needs only one more READ and WRITE operation in comparison with a direct transfer. Moreover, it is done uniformly for every pair of processors, and it takes a time not depending on their distance (or exactly the distance of their numbers).

3. Methods of allotting AU's and buffer memory blocks

For a later purpose to estimate the efficiency of the bus connection mentioned above, it is necessary to give methods of allottnig the AU's and buffer memory blocks in applications for solving partial differential equations in a 3 or 2 dimensional space.

In Fig. 3, V means a cube in a triple number space (i,j,k), corresponding to a discrete coordinate (x_i, y_j, z_k) in a 3 dimensional space (x,y,z). When it is projected to the square OABC being perpendicular to the i-direction, an array of the number points (j,k) is produced on the square as a projection of all the integer lattice points in V. Then, every AU-$((j,k))$ is first made to correspond to each point (j,k) on the square. Such a processor is represented by P in Fig. 3. Processor P is first expected to play the role of computation on every lattice point

which has been projected to the point of P. Such lattice points are represented by a line $\alpha\alpha'$ in Fig. 3.

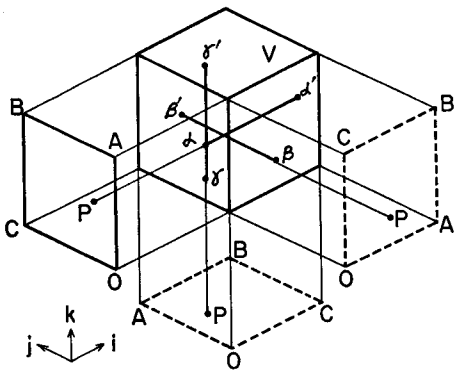

Fig. 3. Correspondence of AU's and buffer memory blocks to a cube of lattice points

Further, the square OABC of the AU's is assumed to stand at two more positions where the projections of V to the j- and k-directions are produced respectively, as shown by the squares with broken sides in Fig. 3. Every AU is also assumed to take the role of computation on the two lines $\beta\beta'$ and $\gamma\gamma'$ of the lattice points. Thus, all the lattice points of V are covered threefold by AU's.

Such a configuration is very favorable especially for the ADI method to solve partial differential equations. It reduces a fundamental cycle of solution to three fractional steps, in each of which a revised difference equation being implicit only in a direction is solved respectively. In order to solve the equation on the concerning fractional step, some results of former fractional steps are usually necessary. Therefore, those results must be transferred from processor to processor.

An aim of setting the special buffer memory is just for such transfers. Fig. 4-a, b, c shows how to correspond the buffer memory to the lattice points in V, and how to connect the buses from the AU's to the buffer memory. The square OABC has the same meaning as in Fig. 3. In Fig. 4-a, the square OLMN is a buffer memory board. Let it be the k-th board from the bottom. Then, as mentioned in the last section, every row of buffer memory blocks, $(i,j)_k$ $(i=0, 1, \cdots, N-1)$, is connected to a corresponding processor $((j,k))$ on the row ab of the square OABC. Also, every column of buffer memory blocks $(i,j)_k (j=0, 1, \cdots, N-1)$ on the same board is connected to a corresponding processor $((k,i))$ on the column $a'b'$. Let every memory block allot to each node of a net produced by the lines drawn from all the processors on ab and $a'b'$ to the i-direction and the j-direction, respectively.

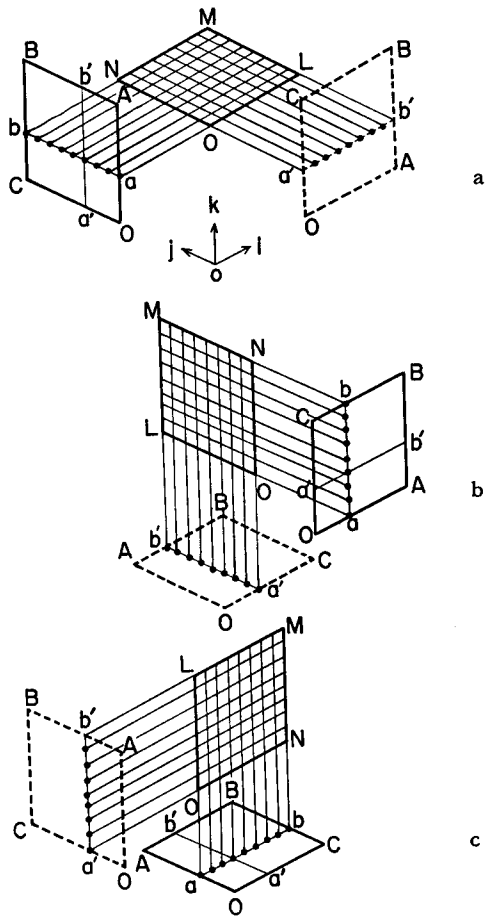

Fig. 4-a a-position of the AU plane and a buffer memory board
Fig. 4-b b-position
Fig. 4-c c-position

Then, the number of processors being related to each buffer memory block is only two. If RAM is used for the buffer memory, two concerning processors are allowed to access a corresponding block directly. On the other hand, if FIFO is used, two DMA channels from two concerning processors are connected to a corresponding block.

As a whole, N buffer memory boards are set up for $k=0, 1, \cdots, N-1$.

As regards the application of the ADI method, after every processor writes some results computed on a corresponding row of lattice points into a corresponding row of buffer memory blocks. It reads other results in the form of a data column from a corresponding column of buffer memory blocks at the position indicated by the square with broken side lines in Fig. 4-a. Then, it goes to the

next step to solve difference equations being implicit with respect to the j-direction.

Fig. 4-b shows the second position of a buffer memory board and the array of processors in the fixed space. Let the board be the k-th one from the left and correspond to the k-th vertical plane of lattice points in V. Fig. 4-b shows also that, once rows of data in the j-direction are written in corresponding rows of buffer memory blocks, the data can be read from columns of buffer memory blocks in the form of data columns in the k-direction.

Fig. 4-c shows the third position of the k-th buffer memory board and the array of processors. Again, it shows that, once rows of data in the k-direction are written in respective rows of buffer memory blocks, the data can be read from columns of buffer memory blocks in the form of data columns in the i-direction.

It must be noted that, though Fig. 4-a, b, c shows different cases of position, the relative relations among the AU's and the buffer memory boards are the same for all cases. Therefore, only one complex of a two dimensional array of AU's and a set of buffer memory boards are sufficient, and they are used in three ways.

By such a consideration, it follows that this computer may reduce the computation time of the ADI method by $1/N^2$ times exactly.

This computer is not restricted only to the application of the ADI method. Later examples of computation will show that it is also efficient for other methods of solution.

Moreover, it is desirable to have an ability to treat difference equations on a two or one dimensional lattice and to get universality. For this purpose, it is preferable that this computer be regarded as another complex of a one dimensional array of processors, and a two dimensional array of buffer memory blocks just like the ADINA Computer I. However, since the ADINA Computer II has not so many buffer memory blocks as to allow a direct transfer of data between any two processors, some proper ideas must be brought in use.

An effective way is brought from the consideration mentioned already, that an indirect transfer is always possible by the help of another processor. In fact, the AU's are all arranged in two ways to the j- and i-directions respectively, with a different ordering as seen in the left and bottom sides of Fig. 5. Then, the given buffer memory blocks are, of course, not sufficient to be allotted to all nodes of a net produced by direct lines drawn from all the AU's in both directions. The nodes to which blocks are really allotted are only what, in Fig. 5, are found in the squares with thick side lines lying diagonal, from right-tiop to left-bottom of the figure. Among those processors being connected to diagonal squares, a direct transfer is always permissible, but it is impossible in general. It is, however,

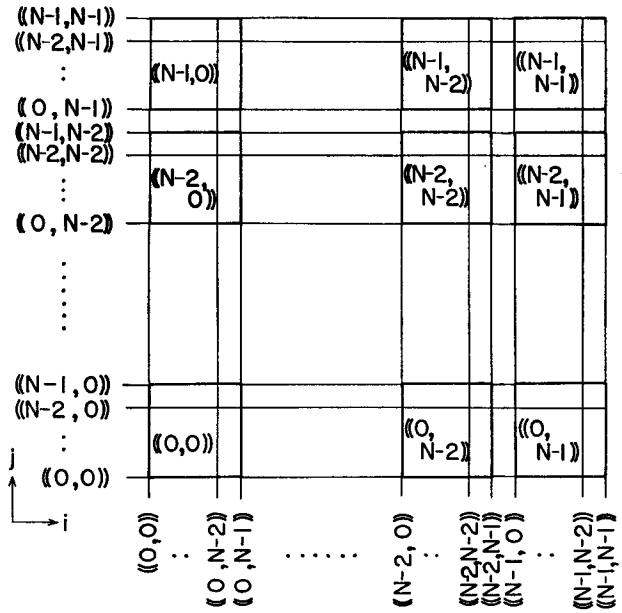

Fig. 5. Two positions of one dimensional array of processors and mediators

allowed to transfer data indirectly between any two processors through another mediating processor indicated in a corresponding square on the off-diagonal. Yet, processors are indicated in the diagonal squares and they also are, if necessary,

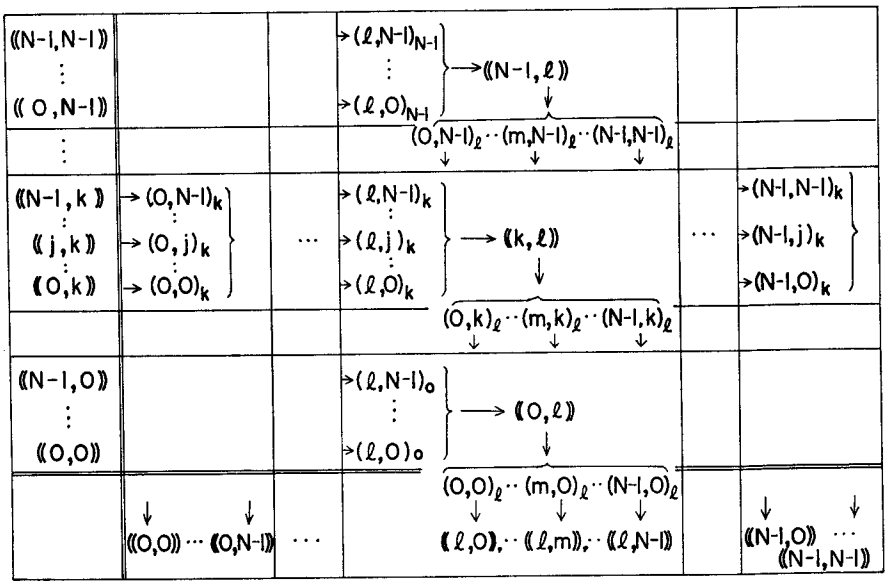

Fig. 6. Data flow through buffer memories and mediators

used to that all transfers among the AU's may be done in a uniform way.

Fig. 6 shows a case of data flow where, it is assumed, every processor has data on each dimensional lattice in the i-direction and then receives data on each dimensional lattice in the j-direction in a fully parallel way. The details are as follows:

i) Every processor $((j,k))$ decomposes N^2 data on each row of lattice points into N sets of N-tuple data, and writes them in N buffer memory blocks $(0,j)_k$, $(1,j)_k, \cdots, (N-1,j)_k$ respectively.

ii) The following procedure is repeated for $m=0, 1, \cdots, N-1$ successively; every processor $((k, l))$ reads the m-th data from all N buffer memory blocks $(l, 0)_k, \cdots, (l,j)_k, \cdots, (l, N-1)_k$ respectively, considers them as a N-tuple data, and writes in the buffer memory block $(m, k)_l$.

iii) Every processor $((l, m))$ reads N sets of N-tuple data from the buffer memory blocks $(m, 0)_l, \cdots, (m, k)_l, \cdots, (m, N-1)_l$ respectively.

It is seen above that, in comparison with a machine having so many buffer memory blocks as to be alloted to all nodes of $N^2 \times N^2$, the second step ii) to mediate data is excessive. However, it does not take so much time. It is hence found that the ADINA Computer II also solves problems on a two or one dimensional lattice with a high efficiency, and has an ability for a wide range of applications.

4. An example of trial manufacture

In this section, a design of manufacturing is exposed in order to give theoretical estimates of efficiency concretely in a later section. Though the aim of developing the ADINA Computer II is clearly an achievement of ultimate high speed by the parallel architecture in addition to the use of ultimate high speed elements, only a trial composed of inexpensive micro-processors on the market is considered, and its efficiency is estimated.

Every processor adopted is a super minicomputer or a LSI minicomputer of TOSBAC series 7, with an ability of 16 bit parallel operations and some macro commands of floating-point number operations.

Provided processors are a MU, 16 SU's and 256 AU's. As the MU, TOSBAC 7/40 is used and, as the SU or AU, TOSBAC 7/10 (micro 7) is used. Some data of those minicomputers are given as follows:

MU has a main memory of NMOS, and its cycle time is 0.36 μs per half word (two bytes), and its maximum capacity is 512 K bytes. Between MU and every SU, a DMA bus is provided, and its transfer speed is 1 M bytes/sec. Every SU

also has a main memory of 48 K bytes (CMOS) and 16 K types (EPROM) with cycle times 1.4 μs and 1.1 μs per half word respectively. A portion of the CMOS memory (16 K bytes) is used as a common memory accessed from both sides of a SU and its corresponding AU. Also, every AU has a main memory of 16 K bytes (CMOS) and 16 K bytes (EPROM). The remainder of the AU's address space is still 32 K tbyes, half of which is used for accessing a common memory, and the other half is used for accessing the buffer memory.

AU's machine times in floating-point operations are listed in Table 1.

Tablle 1. Machine times in floating-point operations by a AU

Operation	Machine time
LE: Register ← (Memory)	15~38.3 μs
STE: Memory ← (Register)	11.25 μs
AE: Register ← (Register) + (Memory)	38.3~58.1 μs
SE: Rdegister ← (Register) + (Memory)	40.13~60 μs
ME: Register ← (Register) * (Memory)	13.1~85.9 μs
DE: Register ← (Register) / (Memory)	15.4~80.6 μs

As the buffer memory, 4096 buffer memory blocks rae prepared, and every block is composed of two chips of MOS static RAM F3539. Its access time is 650 ns and its capacity is 256×8 bits.) Hence, it has a capacity of 256 halfwords.

On manufacturing the buffer memory, 16 memory blocks are set up on a real board in a matrix form of 4×4. Moreover, 16 boards are put together in a 'grand board' which we call a buffer memory board. 16 grand boards complete the buffer memory.

It is seen in Fig. 7 which signals and buses are connected to a real board. In fact, it shows such control signal lines and such address and data buses that come in the buffer moemry block $(0, F)_k$, which is located at the extreme left and upper position on the extreme left and upper board of the k-th grand board.

It is assumed that at any time, either set of lines from the left or that from the bottom of Fig. 7 is selected. The control signals coming to every memory block are $\overline{CS}$(Chip Select), $\overline{OD}$ (Output Disable) and R/$\overline{W}$ (Read/Write selection). $(\overline{P1})_4$ means the fourth bit of the port P1 of AU, and it is an output for selecting all the buffer memory blocks forming a full row on a buffer memory board. $(\overline{P1})_0$ also means the 0-th bit of P1 and is an output for selecting all the blocks forming a full column on another buffer memory board.

Fig. 8 shows the logic for cutting a competition of signals and buses, and also gives a method to access the common memory in a SU. Especially, it gives the logic and the connection among SU-$((k))$, AU-$((j,k))$ $(j=0, 1, \cdots, F)$ and AU-

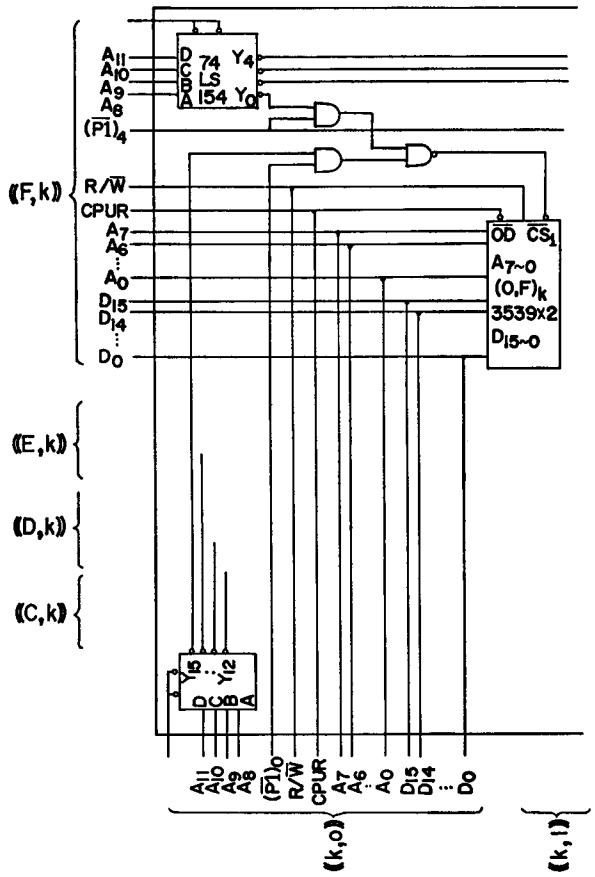

Fig. 7. Signals and data buses coming to a memory board

$((k,i))$ $(i=0,1,\cdots,F)$.

As an illustration of the control, a typical example of application is given as follows:

i) Every $((j,k))$ $(j=0,1,\cdots,F)$ writes some data in all the buffer memory blocks $(i,j)_k$ $(i=0,1,\cdots,F)$, then issues an end signal through $(\overline{\text{PI}})_6$, and closes the horizontal lines of Fig. 7. When they all complete writing, $((k))$ receives a total end signl through $(\overline{\text{PI}})_7$.

ii) $((k))$ clears its $(\overline{\text{PI}})_6$ and issues a singla of having received it through $(\overline{\text{PI}})_7$. The signal comes to every $(\overline{\text{PI}})_7$ of $((j,k))$ $(j=0,1,\cdots,F)$. After receiving it, $((j,k))$ clears its own $(\overline{\text{PI}})_7$. The same signal also comes to every $(\overline{\text{PI}})_1$ of $((k,i))$ $(i=0,1,\cdots,F)$. Aftre receiving it, $((k,i))$ opens vertical lines by making $(\overline{\text{PI}})_2$ low.

iii) Every $((k,i))$ $(i=0,1,\cdots,F)$ writes some results in all the buffer memory

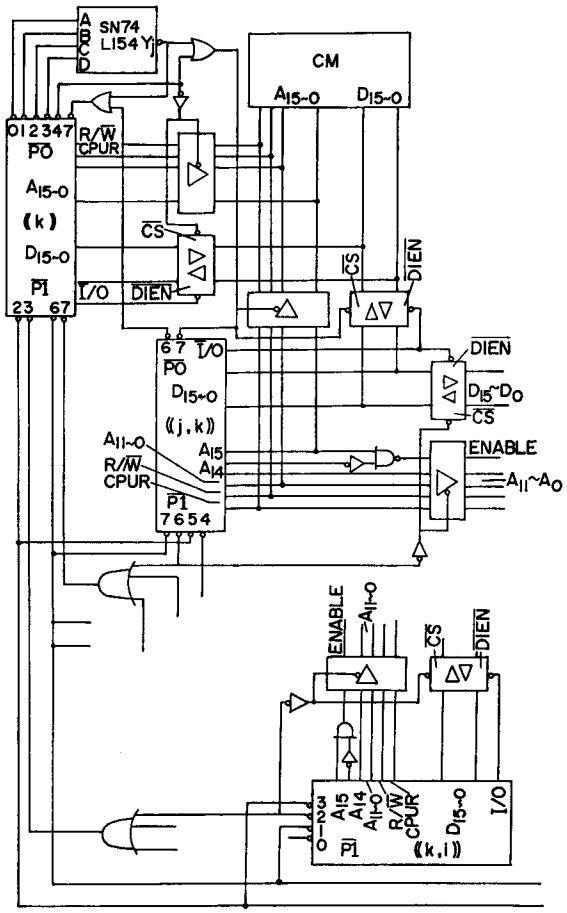

Fig. 8. Logic for help from memory access competition

blocks $(i,j)_k$ $(j=0, 1, \cdots, F)$, then issues another end signal through $(\overline{P1})_2$, and closes the vertical lines. When they all complete writing, $((k))$ receives another total end signal through $(\overline{P1})_3$.

iv) $((k))$ clears $(\overline{P1})_3$ and issue a singla of having received it through $(\overline{P1})_3$. The signal comes to every $(\overline{P1})_3$ of $((k,i))$ $(i=0, 1, \cdots, F)$. After receiving it, $((k,i))$ clears $(\overline{P1})_3$. The same signal also comes to every $(\overline{P1})_5$ of $((j,k))$ $(j=0, 1, \cdots, F)$. After receiving it, $((j,k))$ opens horizontal lines by making $(\overline{P1})_6$ low, and returns to i).

When some results have to be transferred to $((k))$,

v) $((k))$ puts a number '1j' on $(P0)_{4\sim0}$. Then, the lines from $((j,k))$ are opened also to the common memory of $((k))$. After receiving a singla of their having been opened through $(\overline{P0})_7$, $((j,k))$ starts to transfer data.

When it is completed, $((j,k))$ issues an end signal through $(\overline{P0})_2$, and $((k))$ receives it through $(\overline{P0})_7$.

5. Evaluation of efficiency in reference examples of computation

In this section, it is shown by estimating computation time for some examples that the ADINA Computer II has a very high efficiency.

Here two kinds of computation time are compared: a time lost by a parallel processing of the AU's, and that by a uni-processing of a single AU for a computation problem. The former is called the parallel processing time P_t, and the latter is called the uni-processing time S_t. A computation time means here a theoretical value evaluated by following a program written in machine language. In such an evaluation, the times of the floating-point operations are represented by their middle values, since they are not estimated exactly for all cases beforehand. On uni-processing by an AU, it is assumed for a fair comparison that its main memory is enough to keep all the data.

As an index of comparison,

an efficiency R, defined by $R=r/N^2$,

is used when a parallel computation is done by N^2 AU's, where

$$r = S_t/P_t$$

is another index meaning that the parallel processing gets a speed-up by r times in comparison with the uni-processing. The most ideal case is when $r=N^2$, but an ordinary case is when $r<N^2$.

For the convenience of illustrating some methods of application, it is necessary to introduce some symbols and words beforehand.

First, $(m=\cdot)$ indicates a location in a memory block occupied by a concerning data word. It was already mentioned that there are three ways to correspond the buffer memory to a cube in a lattice space as seen in Fig. 4-a, -b, -c. Those positions of the processors and the buffer memory are called a-position, b-position and c-position respectively. In any position, a one dimensional array of N buffer memory blocks projected to a point on the plane of processors in the direction parallel to the side OL, or alternatively to the side ON, is called a memory row or a memory column.

Exampie 1 (ADI method for a 3-dimensional heat conduction)

A difference method is considered here. It is applied for solving the problem of finding a function $u=u(x,y,z,t)$ satisfying the equations

$$\frac{\partial u}{\partial t} = \Delta u \quad \left(\Delta = \frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right), \quad t>0\,, \quad (x,y,z)\in G\,,$$
$$u(x,y,z,t) = g(x,y,z,t)\,, \quad t>0\,, \quad (x,y,z)\in \Gamma\,,$$
$$u(x,y,z,0) = f(x,y,z)\,, \quad (x,y,z)\in G\,,$$

in the domain $G=\{0<x,y,z<1\}$. Here, Γ is the boundary of G, and g and f are the given functions.

G is covered by a net of lattice points produced as cross points of the planes with a parameter h (mesh width)

$$x = x_i = ih\,, \quad y = y_j = jh\,, \quad z = z_k = kh$$
$$i,j,k = 0, 1, \cdots, N+1\,, \quad (N+1)h = 1\,.$$

$U=U_{i,j,k}=U(x_i,y_j,z_k)$ denotes an approximate function for u defined on the lattice. Also, U^n denotes a value of the function U at a time $t=n\tau$ $(n=0,1,2,\cdots)$, τ being a time step.

The heat equation is replaced by the well-known Douglas-Rachford implicit scheme.

$$(1-\lambda\Delta_1)U^{n+1*} = [1+\lambda(\Delta_2+\Delta_3)]U^n\,, \tag{1}$$
$$(1-\lambda\Delta_2)U^{n+1**} = U^{n+1*}-\lambda\Delta_2 U^n\,, \tag{2}$$
$$(1-\lambda\Delta_2)U^{n+1} = U^{n+1**}-\lambda\Delta_3 U^n\,, \tag{3}$$

where $\lambda=\tau/h^2$ and $\Delta_i(i=1,2,3)$ is the difference operator defined as

$$\Delta_1 U_{i,j,k} = U_{i+1,j,k}-2U_{i,j,k}+U_{i-1,j,k}\,,$$
$$\Delta_2 U_{i,j,k} = U_{i,j+1,k}-2U_{i,j,k}+U_{i,j-1,k}$$

and

$$\Delta_3 U_{i,j,k} = U_{i,j,k+1}-2U_{i,j,k}+U_{i,j,k-1}$$

respectively. The above scheme determines U^{n+1} from U^n, and it is used repeatedly for $n=0,1,2,\cdots$. It is seen in the scheme that two functions, U^{n+1*} and U^{n+1**}, stand as auxiliary functions. Therefore, three fractional steps of finding U^{n+1*}, U^{n+1**} and U^{n+1} complete one step from n to $n+1$. An important characteristic of the scheme is that every equation of (1), (2) and (3) has a fully blocked coefficient matrix. In fact, (1) may be decomposed in N^2 subsets corresponding to all (j,k)'s respectively, each of which has only N unknows $U^{n+1*}_{i,j,k}(i=1,2,\cdots,N)$. Therefore, they are solved independently of one another. (2) and (3) may also be decomposed in a similar way. Also, every coefficient matrix is tri-diagonal. Hence, every equation is easily solved by the so-called double sweep method, in which the number of arithmetic operations amounts to $O(N)$ in general.

The ADINA Computer II is very suitable for such a problem. In fact, if the

number N of lattice points standing on a line parallel to each coordinate axis is assumed to be equal to the number of processors on each row or column of the AU's plane for simplicity, every subsystem of (1) is solved by a corresponding processor respectively. Further the (j,k)-th equation is simultaneously solved by $((j-1,\ k-1))$. Then, the N^2 subsystems are all solved in a fully parallel way. Equations (2) and (3) are also solved in a similar manner.

Details of the procedure are as follows:

i) First, it is assumed that the buffer memory in the c-position has $U^n(m=0)$, $\lambda\Delta_2U^n(m=1)$ and $\lambda\Delta_3U^n(m=2)$, and that in the b-position it also has $\lambda\Delta_3U^n(m=3)$.

ii) Every processor gets U^n, $\lambda\Delta_2U^n$ and $\lambda\Delta_3U^n$ from each of the memory rows $(m=0, 1, 2)$, and then gets each component of the right hand vector of (1). Every processor solves each equation, and U^{n+1*} is then found. Further, U^{n+1*} and $\lambda\Delta_2U^n$ are stored in the memory rows $(m=4, 5)$ of the a-position.

iii) Every processor gets U^{n+1*} and $\lambda\Delta_2U^n$ from the memory columns of the a-position, solves each equation of (2), and writes its solution U^{n+1**} in the memory row $(m=6)$ of the b-position.

iv) Every processor gets U^{n+1**} and $\lambda\Delta_3U^n$ from the memory columns $(m=6, 3)$ of the b-position, solves each equation of (3). It then stores its solution U^{n+1} in the memory column $(m=6)$ of the b-position, and also in the memory row $(m=0)$ of the c-position.

v) Every processor gets U^{n+1} from the memory column $(m=6)$ of the b-position, finds $\lambda\Delta_2U^{n+1}$, and stores it in the memory row $(m=3)$ of the b-position. Moreover, every processor gets U^{n+1} and $\lambda\Delta_2U^{n+1}$ from the memory columns $(m=6, 3)$ of the b-position, and stores them in the memory rows $(m=0, 1)$ of the c-position. Finally, every processor gets U^{n+1} from the memory row $(m=0)$ of the c-position, computes $\lambda\Delta_3U^{n+1}$, and stores it in the memory row $(m=2)$.

Then, the procedure returns to i) of the next step.

In this case, it is found, without summing up the computation times, that the parallel processing time P_t is just $1/N^2$ times of the uni-processing time S_t, that is, the efficiency is 100%. It is because all the work of the AU's is done in a fully parallel way, and without any special additional transferring.

Example 2 (A simple difference scheme for the problem of Example 1)

A simple explicit shceme

$$U^{n+1} = U^n+\lambda(\Delta_1U^n+\Delta_2U^n+\Delta_2U^n)$$

is considered. A realization of this scheme on the ADINA Computer II is as follows:

i) First, it is assumed that U^n is stored in the buffer memory ($m=0$) of the a-position.

ii) Every processor gets U^n from the memory row ($m=0$) of the a-position, computes $\Delta_1 U^n$, and stores it in the memory row ($m=1$) of the same position.

iii) Every processor gets U^n from the memory column ($m=0$) of the a-position, computes $\Delta_2 U^n$, and stores it in the memory column ($m=2$).

iv) Every processor gets U^n from the memory row ($m=0$) of the a-position, and stores it in the memory column ($m=3$) of the c-position.

v) Every processor gets U^n from the memory row of the c-position, computes $\Delta_3 U^n$, and stores it in the memory row ($m=4$) of the c-position.

vi) Every processor gets $\Delta_3 U^n$ from the memory column ($m=4$) of the c-position, and gets U^n, $\Delta_1 U^n$ and $\Delta_2 U^n$ from the memory rows ($m=0, 1, 2$) of the a-position,. It then finds U^{n+1}, and stores it in the memory row ($m=0$) of the a-position.

Then, the procedure returns to i) of the next step.

In this case, it is found that

$$R = \left(1+\frac{\text{LE}+\text{STE}}{7\text{LE}+4\text{ME}+6\text{AE}+3\text{SE}+7\text{STE}}\right)^{-1}$$
$$= 0.96 .$$

Here, LE, AE and etc. represent computation times of the floating-point operations, to which some middle values of those mentioned in the beginning of the last section are given.

The above two examples treat the problem on a 3-dimensional lattice in such a way that every lattice point corresponds to each buffer memory block. Such a problem is directly fitted for this machine, and is consequently solved with a very high efficiency.

There are, of course, some problems for which it is desirable that the AU's be arranged in a one-dimensional array. In such cases, the array is usually assumed to take two positions of the left vertical side and the bottom horizontal side, as seen in Fig. 5. These positions are called A-position and B-position respectively. Since the buffer memory blocks are not enough to correspond to all nodes of the net by the lines drawn from all places of the processors on the vertical and horizontal sides, it is better to provide a portion of the buffer memory and private memories for exclusive use.

It is assumed that every processor is provided with two one-dimensional arrays for exclusive use, which correspond to the i-th and j-th direction, and have N memory blocks, respectively. Those arrays are again called the memory row and the memory column. Notice that 'block', 'memory row' and 'memory column' are different from the former ones. Every memory block has a capacity of words. A location in every memory block is specified by $(m=\cdot)$.

Example 3 (ADI method for a two-dimensional heat conduction)

The problem is to find $u=u(x, y, t)$, such that

$$\frac{\partial u}{\partial t}=\Delta u \quad \left(\Delta=\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right), \quad t>0\,, \quad (x, y)\in G\,,$$
$$u(x, y, t)=g(x, y, t)\,, \quad (x, y)\in \Gamma\,,$$
$$u(x, y, 0)=f(x, y)\,, \quad (x, y)\in G\,,$$

where $G=\{0<x, y<1\}$, Γ is its boundary, and f, g are the given functions.

A net of computation points is produced by the vertical and horizontal lines

$$x=x_i=ih\,, \quad y=y_j=jh\,,$$
$$i, j=0, 1, \cdots, N^2+1\,, \quad (N^2+1)h=1\,.$$

Other symbols appearing in the following are assumed to have the same meanings as in the former examples.

The heat equation is replaced by the following ADI scheme:

$$(1-\lambda\Delta_1)U^{n+1*}=(1+\lambda\Delta_2)U^n\,, \tag{1}$$
$$(1-\lambda\Delta_2)U^{n+1}=2U^{n+1*}-(1+\lambda\Delta_2)U^n\,, \tag{2}$$

where $\lambda=\tau/2h^2$.

It is here clear that one step from n to $n+1$ takes two fractional steps to find U^{n+1*} from (1) and U^{n+1} from (2). These computations are allotted to all AU's in the following way: the j-th processor from the bottom of the A-position plays the role of computation on the lattice points on the horizontal line $y=y_j$, and the i-th processor from the left in the B-position plays the role of computation on the lattice points on the line $x=x_i$. Therefore, every processor bears the computation on two one-dimensional arrays of lattice ponits.

Details of solving (1) and (2) are as follows:

i) First, it is assumed that $(1+\lambda\Delta_2)U^n$ is stored in the whole memory columns $(m=0)$.

ii) It is transferred to the whole memory rows $(m=0)$ in a direction contrary to that shown in Fig. 6. The processor $((l, m))$, for example, divides the data of its own memory column into N parts, and writes them into N buffer

memory blocks (in the former meaning) $(m,0)_l, \cdots, (m,k)_l, \cdots, (m,N-1)_l$ respectively. The processor $((k,l))$ then reads the data in $(0,k)_l, \cdots, (m,k)_l, \cdots, (N-1,k)_l$, edits them in the form corresponding to the memory row, and writes in $(l,0)_k, \cdots, (l,N-1)_k$. Further the processor $((j,k))$ collects data blocks from $(0,j)_k, \cdots, (N-1,j)_k$, and writes them in its own memory row.

iii) Every processor finds U^{n+1*} from (1), and writes it in its memory row $(m=1)$. These data in the memory rows $(m=1)$ are then transferred to the whole memory columns $(m=1)$ in the manner shown in Fig. 6.

iv) Every processor reads these data from its own memory columns $(m=0, 1)$, solves (2), and writes its solution U^{n+1} in its memory column $(m=2)$. Further, every processor computes $(1+\lambda\Delta_2)U^{n+1}$, and stores it in its memory column $(m=0)$.

Then, the procedure returns to i).

In this case, it is found that

$$\mathrm{R} = \left(1+\frac{6(\mathrm{LE}+\mathrm{STE})}{10\mathrm{LE}+8\mathrm{ME}+8\mathrm{AE}+4\mathrm{DE}+\mathrm{SE}+10\mathrm{STE}}\right)^{-1} = 0.86\,.$$

The numerator 6(LE+STE) is the exess time for transferring data. It causes a fall in efficiency, but still, 0.86 means a high efficiency.

Example 4 (A simple explicit difference scheme for the problem of Example 3)

Here, a simple explicit scheme

$$U^{n+1} = U^n+\lambda\Delta_1(U^n+\Delta_2U^n)\,, \quad \lambda = \tau/h^2$$

is considered. A procedure of using it is given as follows:

i) First, it is assumed that U^n is stored in the whole memory rows $(m=0)$.

ii) U^n in the memory rows $(m=0)$ is transferred to the whole memory columns $(m=0)$ in the manner shown in Fig. 6.

iii) Every processor reads U^n from its own memory column $(m=0)$, computes Δ_2U^n, and sends it to the memory row $(m=1)$ in the inverse manner of Fig. 6.

iv) Every processor reads data from its own memory rows $(m=0, 1)$, finds U^{n+1}, and stores it in the memory row $(m=0)$.

Then, the procedure returns to i).

In this case,

$$R = \left(1+\frac{6(LE+STE)}{4LE+3ME+4AE+2SE+4STE}\right)^{-1}$$
$$= 0.72\,.$$

Example 5 (Product of two matrices A and B with the order of N^2)

It is assumed that every row vector $\vec{a}^{(p)}$ of A is stored in a memory row of each correspondnig processor in the A-position, and that every column vector $\downarrow b^{(q)}$ of B is stored in a memory column of each corresponding processor in the B-position.

The multiplication is done in the following manner. First, $\downarrow b^{(0)}$ is transferred to all processors. After receiving it, every processor computes a corresponding scalar product $\vec{a}^{(p)}\cdot\downarrow b^{(0)}$. Such computation is repeated successively for every $\downarrow b^{(1)}, \cdots, \downarrow b^{(N^2)}$. It must be here noted that every buffer memory block may contain only $N^2/2$ words at most, while every row and column vector has N^2 words (N^2=256). Therefore, every computation of $\vec{a}^{(p)}\cdot\downarrow b^{(q)}$ must be divided in to two parts of $\vec{a}_F^{(p)}\cdot\downarrow b_F^{(q)}$ and $\vec{a}_L^{(p)}\cdot\downarrow b_L^{(q)}$, and then be completed by summing them up, where the suffixes F and L mean the former and latter half of every vector, respectively.

Details of the above procedure are given as follows;

The following procedure is repeated for l, m=0, 1, $\cdots$, $N-1$ successively:

i) A processor $((l,m))$ writes its own vector $\downarrow b_F$ on all buffer memory blocks $(m,k)_l$ (k=0, 1, $\cdots$, $N-1$).

ii) Every processor $((k,l))$ (k=0, 1, $\cdots$, $N-1$) reads it from $(m,k)_l$, and writes it in all $(l,j)_k$ (j=0, 1, $\cdots$, $N-1$).

iii) Every processor $((j,k))$ reads it from each corresponding block $(l,j)_k$, and computes each $\vec{a}_F\cdot\downarrow b_F$.

iv) In a similar manner, $\vec{a}_L\cdot\downarrow b_L$ is also computed, and is added to the former part of the products.

In this case,

$$R = \left(1+\frac{2(LE+STE)}{LE+ME+AE+STE)}\right)^{-1}$$
$$= 0.64.$$

Therefore, the speed of parallel processing is only $0.64N^2$ times that of uni-processing. It is because an essential part of parallel computation (which is described by the denominator of the expression) is so simple and short that the time used for transferring (which is described by the numerator of the same expression) is relatively large.

The cases of Examples 3, 4 and 5 can be treated more efficiently by the ADINA

Computer I, but its hardware becomes very bulky. Therefore, in the case of a large number of processors, the ADINA Computer II seems to be superior to the ADINA Computer I for practical use.

Reference

1) Nogi, T. and Kubo, M.: ADINA Computer I (Alternating Direction Immediate Nexus Array Computer I) I. Architecture and Theoretical estimates, Memoirs of the Faculty of Engineering, Kyoto University, Vol. XLII, Part 4, Oct. 1980.