ADENA Computer III

AUTHOR(S):

NOGI, Tatsuo

CITATION:

NOGI, Tatsuo. ADENA Computer III. Memoirs of the Faculty of Engineering, Kyoto University 1989, 51(2): 135-152

ISSUE DATE:

1989-04-30

URL:

http://hdl.handle.net/2433/281395

Mem. Fac. Eng., Kyoto Univ. Vol. 51, No 2 (1989)

ADENA Computer III

By

Tatsuo NOGI

(Received December 26, 1988)

Abstract

This paper gives a proposal about a new parallel architecture composed of many vector processors, and being especially designed for number-crunching simulation problems in engineering and science. This computer is elaborated to systematize up-to-date supercomputer units (vector processors) in a united parallel machine of the SIMD type. It also aims to reduce hardware quantity of communication network among so many processors as in ADENA I and II, proposed previously by the author.

1. Introduction

This paper proposes a new parallel computer of the SIMD type, especially designed for computer simulation in science and engineering. Up to now, several kinds of multiprocessor systems have been proposed or developed especially for the solution of partial differential equations. Their representatives are ILLIAC-IV[1], DAP[2] and PAX[3]. They all have a two dimensional array of processors, essentially connected between each pair of neighbouring ones. Hence, they have a common defect in the ability of transferring data among separate processors. Therefore, some new networks of data paths are proposed to make it possible to allow various ways of transferring. The author also has proposed new architectures ADENA I and ADENA II (renamed from previous ADINA I and ADINA II, [4]-[8]). They have buffer memory arrays as to process universal algorithms for multidimensional simulation problems, so called, splitting-up operator methods (including ADI method). Especially, ADENA II promises practical machines, and it has been really developed by collaboration between Kyoto University (the author) and the Matsushita Electric Industrial Company Ltd.. The more processors there are, the more buses may be lengthened, and the speed-down of data transfer may be induced.

On the other hand, the vector pipeline architecture has been fully developed as a natural extension of universal computers of the ordinary type, and makes the main cur-

Department of Applied Mathematics and physics

rent of the supercomputer. Nowdays, such vector pipeline architecture comes up to the physical limit of performance. Further development is hopeless. From this point of view, parallel architecture is again hoped for.

Our ideas is, hence, to combine the ADENA parallel architecture and the vector pipeline archtecture so that it may reduce hardware quantity in the previous ADENA architecture on one hand, and may give a natural way of operating many vector pipeline processors in parallel on the other hand. We call such a machine a parallel vector computer or ADENA Ⅲ. This new architecture, we hope, may promise a next generation of supercomputers.

2. Basic Architecture of ADENA Ⅲ

2. 1 Our parallel vector computer (ADENA Ⅲ) is used in such a way of being attached to a universal front-end computer. All input and output devices, memory disks and consort terminal are connected to such a front-end computer, which plays a role of data input and output, compiling and controlling the parallel vector computer.

We give two categories to our parallel vector computer : a single block and a complex of several blocks. The former computer has a control part CB composed of a control processor and its main memory CM, a processor part PB composed of the number N (a definite integer number) of parallel processing vector processors PU[r], r $=1,2,...,N$ and the same number of scalar processors SU[r], $r=1,2,...,N$, with scalar memory storages PM[r], $r=1,2,...,N$, respectively. There is also a main memory bank part MB forming a cubic array of the number M^3 ($M=pN$, p is an integer) of memory units, $\{MU[i,j,k],\ i,j,k=1,2,...,M\}$.

Next, the complex system is as follows : Taking such a single processor part PB and such a memory bank part MB as respective units, we form an array of L^3 memory bank parts, $\{MB[I,J,K],\ I,J,K=1,2,...,L\}$. We call the complex the (L^2,L^3)— parallel vector computer (or simply the (L^2,L^3)— system). We also call the former single block the $(1^2,1^3)$— system.

2. 2 We will first explain the $(1^2,1^3)$— system. Its control processor CU decodes object codes stored in CM, successively, and gives a same command sequence to all vector processors. By this command sequence, all processors PU[r], $r=1,2,...,N$ run in a fully synchronized mode. Each vector processor has a number of vector registers and vector pipeline processing units, in addition to a scalar register and a scalar processing unit. It has a private memory PM, as a scalar data storage, while it has a main memory bank MB for vector data. The bank part MB is an array of memory units with same

capacity, $\{MU[i,j,k],\ i,j,k=1,2,...,M\}$. Each processor PU[r] can have access to respective two-dimensional subarrays $\{MU[i,j,\bar{k}_q],\ i,j=1,2,...,M\}$, $q=1,2,...,p$, where $\bar{k}_q=p(r-1)+q$, via a set of high way 'row' buses.

By the so called interleaved way, it has access to respective sets of memory banks:

$$\{MU[i,1,\bar{k}_1]-MU[i,2,\bar{k}_1]-.....-MU[i,M,\bar{k}_1]-$$
$$-MU[i,1,\bar{k}_2]-MU[i,2,\bar{k}_2]-.....-MU[i,M,\bar{k}_2]-$$
$$.....$$
$$-MU[i,1,\bar{k}_p]-MU[i,2,\bar{k}_p]-.....-MU[i,M,\bar{k}_p]\}$$
$$(i=1,2,...,M)$$

, that is, it reads or writes a vector with length pN, whose elements are at a respective memory location with the same address as the above sequence of memory banks. In addition, the same set of vector processors also can have access to the same main memory banks MB here via another set of high way 'column' buses. Each processor PU[r] $(r=1,2,...,N)$ is connected to respective two-dimensional subarrays $\{MU[\bar{i}_q,j,k],\ j,k=1,2,...,M\}$, $q=1,2,...,p$, where $\bar{i}_q=p(r-1)+q$. Hence, it can have access to respective sets of memory banks in the interleaved way:

$$\{MU[\bar{i}_1,j,1]-MU[\bar{i}_1,j,2]-.....-MU[\bar{i}_1,j,M]-$$
$$-MU[\bar{i}_2,j,1]-MU[\bar{i}_2,j,2]-.....-MU[\bar{i}_2,j,M]-$$
$$.....$$
$$-MU[\bar{i}_p,j,1]-MU[\bar{i}_p,j,2]-.....-MU[\bar{i}_p,j,M]\}$$
$$(j=1,2,...,M)$$

, that is, it reads or writes a vector with length pN, whose elements are at a respective memory location with the same address as the above sequence of memory banks.

2. 3 Now we are going to explain the architecture of a general (L^2,L^3) system. This may open a way to enlarge the $(1^2,1^3)$ system. Making such a processor part PB (1^2) as a unit, we take a square array of L^2 units, $\{PB[J,K],\ J,K=1,2,...,L\}$, and further, making such a main memory block MB (1^3) as a unit, we take a cubic array of L^3 units, $\{MB[I,J,K],\ I,J,K=1,2,...,L\}$. Every control processor CU on each PB decodes common object codes and drives corresponding PU's in a synchronous way all over the total system. Every PB[J,K] may have access to each corresponding row of main memory blocks, $\{MB[1,J,K],MB[2,J,K],...,MB[L,J,K]\}$. Its access manner is the same as that via the row bus network in the $(1^2,1^3)$ system. An essential point is that the row buses are common through the respective rows of MB's.

In consequence, each group of L^2 vector processors (one from each PB) can have access

to a L^2 times length vector (pM word length by a vector processor) concurrently and hence to all L^2NpM word lengths, as a whole.

In addition, every PB [K,I] may have access to each corresonding column of main memory blocks, {MB [I,1,K], MB [I,2,K], ..., MB [I,L,K]}. Its access manner is again the same as that via the column bus network in the $(1^2,1^3)$ system. As above, it is important that the column buses are common through the respective columns of MB's. In consequence, all L^2N vector prcessors may process a vector of an L^2NpM word length. A typical application of any (L^2,L^3)— system is as follows: one or more PB's have access to main vector banks via row buses, (we call then each set of such banks a bank row), and process those vectors concurrently (row operation), or alternatively access the same banks via column buses (a bank column) and process those vectors concurrently (column operation), and repeat any sequence of row and column operations.

3. Fundamental Data Structure

3. 1 For an illustration of processing, we will introduce a data structure suitable for ADENA III. Our machine has special characteristics in array data and a way to store them in main memory banks. For simplicity, we consider a $(1^2,1^3)$ system with $p=1$. We further suppose that the number of PU's is N, and the memory bank (cubic) array is of $N\times N\times N$ units. Consider a typical three-dimensional array U, which might be declared as

REAL U(N,N,N)

in a usual computer. How is it to store such data {U(I,J,K), I,J,K=1,N} in our main memory banks? It may be natural to hold every element U(I,J,K) at a location with a same address (depth) in each corresponding bank MU[i,j,k]. There are three plausible ways of such correspondence, in which the 'k' axis fixed to the memory bank hardware is assigned to anyone of the physical data axis K, J and I. We name such an assignment a allocation, b allocation and c allocation respectively. According to these three allocation ways, we give different expessions of data:

a (k axis being parallel to K axis): U(I,J,/K/)
b (k axis being parallel to J axis): U(I,/J/,K)
c (k axis being parallel to I axis): U(/I/,J,K)

These expressions are those for stored data, and we call them 'stock expression'. They must be, we consider, stored differently one another, and hence must be declared independently. Needless to say, only necessary data are specified. Suppose, for example, that we use first two kinds of data. We then declare them as follows:

REAL U(N, N, /N/), U(N, /N/, N)

As mentioned in the last section, our machine can process those data with having access to memory banks via row buses or column buses, alternatingly. We here give different data expressions according to such access ways, which we call 'access expressions'. To each a, b, and c, there correspond two access expressions, and hence we have altogether 6 kinds of access expressions as seen in the following table:

Allocation	stock expression	Access expression	
		row access	column access
a	U(I,J,/K/)	U(,J,/K/)(I)	U(I,,/K/)(J)
b	U(I,/J/,K)	U(I,/J/,)(K)	U(,/J/,K)(I)
c	U(/I/,J,K)	U(/I/,,K)(J)	U(/I/,J,)(K)

Our machine allows one to edit easily a stock array into another stock array. This ability is essential in using this machine. In replacing |U(I,J,/K/)| by |U(I,/J/,K)|, for example, its procedure is as follows: each processor first reads by the column access and then writes by the row access, illustrated as:

$$|U(I,/J/,K)| \xrightarrow{\text{column access}} |U(,/J/,K)(I)| \text{ (processor)} \xrightarrow{\text{row access}} |U(I,J,/K/)|$$

In replacing |U(I,/J/,K)| by |U(/I/,J,K)|,

$$|U(/I/,J,K)| \xrightarrow{\text{column access}} |U(/I/,J,)(K)| \text{ (processor)} \xrightarrow{\text{row access}} |U(I,/J/,)(K)|$$

In replacing |U(/I/,J,K)| by |U(I,J,/K/)|,

$$|U(I,J,/,K/)| \xrightarrow{\text{column access}} |U(I,,/K/)(J)| \text{ (processor)} \xrightarrow{\text{row access}} |U(/I/,,K)(J)|$$

Naturally, their inverse procedures also are possible.

Fig. 1 (a), (b) and (c) shows a schematic diagram of a, b and c allocations of our parallel vector computer. In each diagram, a central cube is the main memory bank MB. We designate three axes attached to the physical main memory block by i-axis, j-axis and k-axis, respectively. For an illustration, the plane vertical to k-axis is hatched. Two squares ABCD are arrays of vector processors, one being of row access and anoter being of column access (only one ABCD exists physically). Every parallel segment on ABCD illustrates each vector processor PU. Processors of segments parallel to the (i,j)-plane are of row access, and those of segments parallel to the (j,k)-plane are of column access. For array data U's, their access expressions also are given near the corresponding diagrams.

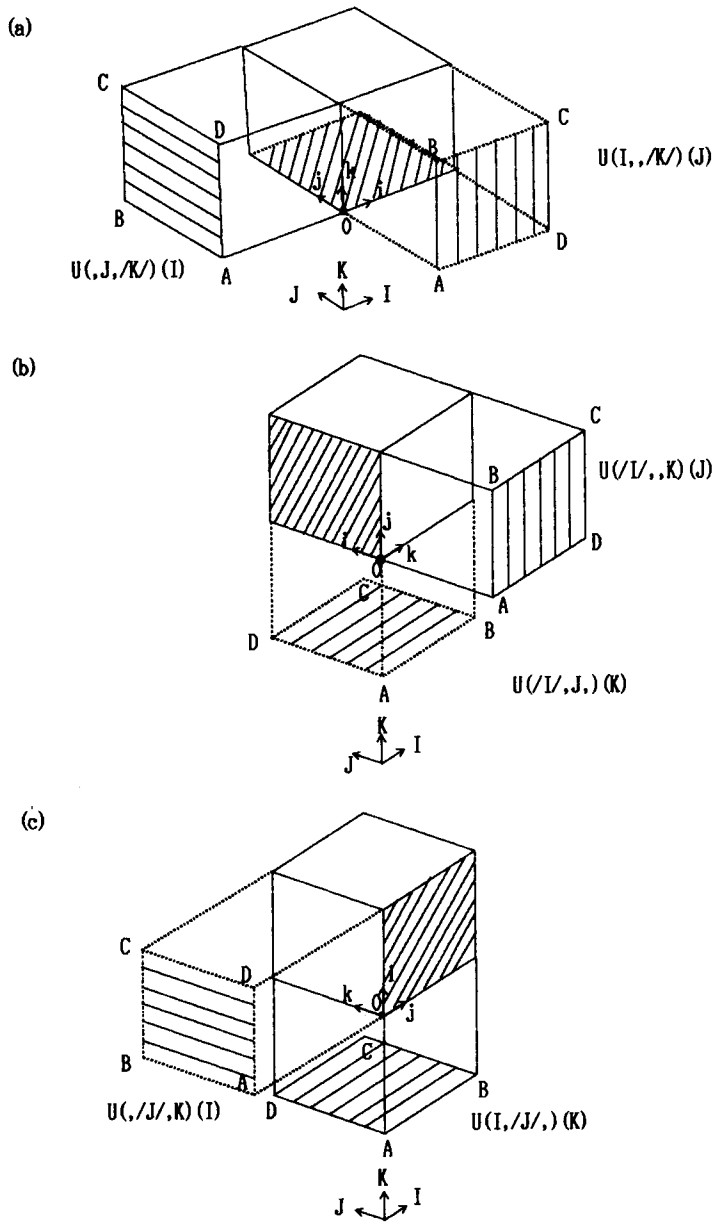


Fig. 1 (a)(b)(c) Three allocation ways of ADENA III.

3. 2 We shall give some examples of parallel statements. We will use ADETRAN III language specially designed for ADENA III, which is FORTRAN-like.

```
          PDO     K=1,N
            PIPE  J=1,N
(I)           DO  10  I=2,N-1
       10       V(,J,/K/)(I)=U(,J,/K/)(I-1)+U(,J,/K/)(I+1)
            PEND  J
          PEND  K
```

or simply

```
          PDO     J,K=1,N   *(,J,/K/)
(I)'        DO  10  I=2,N-1
       10       V(I)=U(I-1)+U(I+1)
          PEND
```

It is an example of parallel processing in row access. Every procesor with each number K processes, in parallel, its sharing vector (its components ordered with index J) sucessively with increasing I as commanded in the DO paragraph. Naturally, PDO paragraph (composed of statements between a PDO statement and its next PEND statement) must have the same indexes in former parentheses (such index being only a variable, not an expression). This restriction rather allows such simple statements as seen in (I)'. In the latter parentheses, expressions of index variables are admitted.

The last example is for the case of allowing expression in the lst index. To allow expression in the 2nd index, b allocation stock data are used:

```
          PDO     K,I=1,N
          DO  20  J=2,N-1
(II)  20  W(I,,/K/)(J)=V(I,,/K/)(J+1)-V(I,,/K/)(J-1)
          PEND
```

Now, we can make (II) to follow (I)' immediately. In this case, the machine itself only has to switch from row access to column access. But, if it were desired to continue such corresponding to that in slash pair in the preceeding statements, data edition would be necesary. After (I)', we put such data edition statements as

```
          PASS      I,J,K=1,N
            V(I,/J/,K)=V(I,J,/K/)
          PEND
```

to get b allocation stock data |V(I,/J/,K)|, and we can then continue the following:

```
          PDO  I,J=1,N
            DO  30    K=2,N-1
```

```
30        Z(I,/J/,)(K)=V(I,/J/,)(K+1)+V(I,/J/,)(K-1)
      PEND
```

Or, we put

```
      PASS I,J,K=1,N
        V(/I/,J,K)=V(I,J,/K/)
      PEND
```

to get c allocation stock data $\{V(/I/,J,K)\}$, and we can then continue

```
      PDO  I,J=1,N
        DO  40  K=2,N-1
40        Z(/I/,J,)(K)=V(/I/,J,)(K+1)+V(/I/,J,)(K-1)
      PEND
```

In the above examples, the size of the data array ($N\times N\times N$) coincided with that of the main memory banks and hence, the $(1^2,1^3)$— system was sufficient. In general, however, such is not the case.

Any one dimension may have a size of less or more than N. In a small size case, mask operation or short-length-vector processing is necessary. However, in a large size case, main memory banks are used in multiple (those having a same index modulo N are stocked in different locations in each corresponding memory bank). The set of vector processors also must be used in multiple (one fold by one fold, sequentially). Admitting vast main memory banks with $p>1$, it is better to reduce the multiple use of memory banks and practice the long-length-vector (pN) processing. Further, in order to increase parallel processing, we can use an extended (L^2,L^3) system.

3. 3 Our machine is able to process also two-dimensional array data. It is realized by mapping those data into four-dimensional array data. We will illustrate it by a simple example. Suppose for simplicity that we have a $(1^2,1^3)$— system with $p=1$, and the number of PU's is N, and consider data array that might be declared in ordinary fashion as

```
REAL U(N**2, N**2).
```

In this case, there correspond two kinds of allocation and stock expressions. Both allocations allow only a kind of access expression respectively as shown in the following table:

It is necessary to distribute this two-dimensional array in two-dimensional memory banks array. Selecting a fixed pair of allocations, for example, a and c allocations, we will give the following index mapping: writing index I and J as

Allocation	Stock expression	Access expression	
		row access	column access
a′	U(I,/J/)	U(,/J/)(I)	------
b′	U(/I/,J)	------	U(/I/,)(J)

$$I=P*N+I',\ J=Q*N+J'$$

we assume the index pair (P,I′) and (Q,P′) instead of I and J, and access expression U(,J′/Q/)(I) and U(/P/,,I′)(J) instead of U(,/J/)(I) and U(/I/,)(J), respectively. Data edition

```
PASS   I,J=1,N**2
U(/I/,J)=U(I,/J/)
PEND
```

is, then, realized as follows: considering $\{U(,J',/Q/)(I)\}$ as a row access array of 4-dimension $\{U(,J',/Q/)(I')(P)\}$ (P corresponds to multipicity index). Rereading further, it is a column access array $\{U(I',,/Q/)(J')(P)\}$, which is again edited as

$$\{U(I',,/Q/)(J')(P)\} \longrightarrow \{U(/P/,,I')(J')(Q)\}.$$

The last edition demands ordering skipped components by N (corresponding to increment of P) into a sequence.

4. Implementation design

4. 1 In this section, we will give some basic designs for implementation of our architecture. Fig. 2 shows an overall feature of ADEMA-II. The part enclosed by the dotted lines is the parallel vector computer, which is connected to a universal computer for front / end processing, that plays roles of data I / O or compiling.

The parallel vector computer is, as a basic model, composed of the control part CB, the processing part PB, and the main memory part MB. The control part CB is connected to the front / end computer via data buses and some signal lines. It has a control processor CU and a program memory CM, in which program codes produced by the front / end computer and transferred are stored. CU reads such program codes sequencially from CM, decodes them and produces control signals for itself and overall PB. It also plays a role of data transmission between the front / end computer and PB.

The processing part PB has a number of PU's which is composed of a scalar processing unit with a local scalar memory and a vector processing unit. Each scalar unit

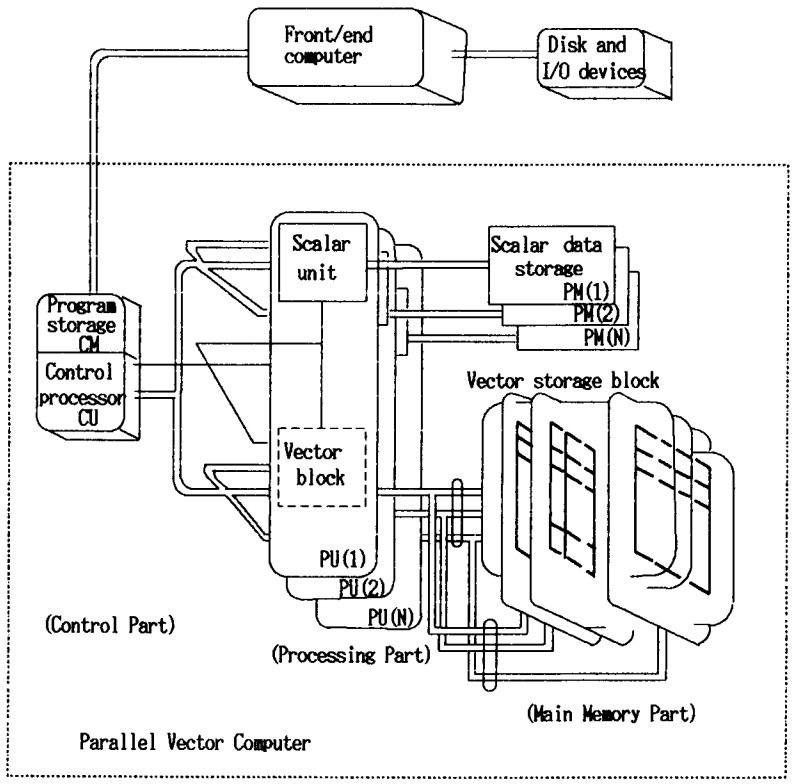


Fig. 2 Whole feature of ADENA III.

executes scalar operations, while each vector unit does vector pipeline processing. Vector units form a complex of parallel processing altogether with the main memory block MB.

In processing, all scalar units and vector units are fully synchronized under control of CU (SIMD type.) Each unit corresponds only to its related instructions. The main control part of our machine is a parallel vector processing, and we mention this part only in the following.

Fig. 3 shows an outline of a vector unit of PU. (Simply, we call it a vector unit PU.) PU gets instructions from CU in its instruction register, and drives the vector professing part and DMA controller by its inner control signals. PU has, as outward lines to CU, a data bus, an instruction bus and a unit selecting address bus, while as inward lines to MB, data buses and address buses.

Fig. 4 illustrates a scheme of connecting a PU with a subarray of the memory bank unit MU's. Each MU is here standing at a node of lattice. Suppose that each r-th

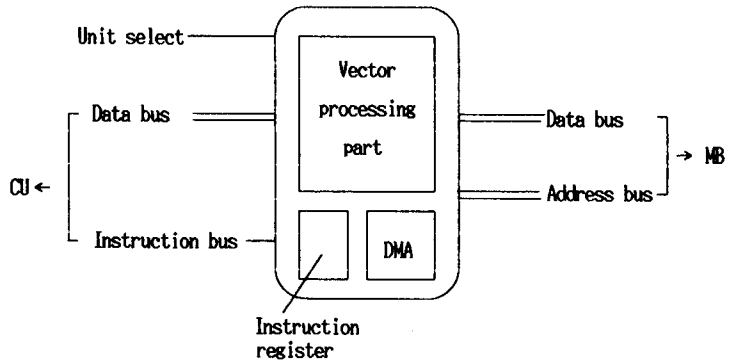


Fig. 3 A single vector unit of PU.

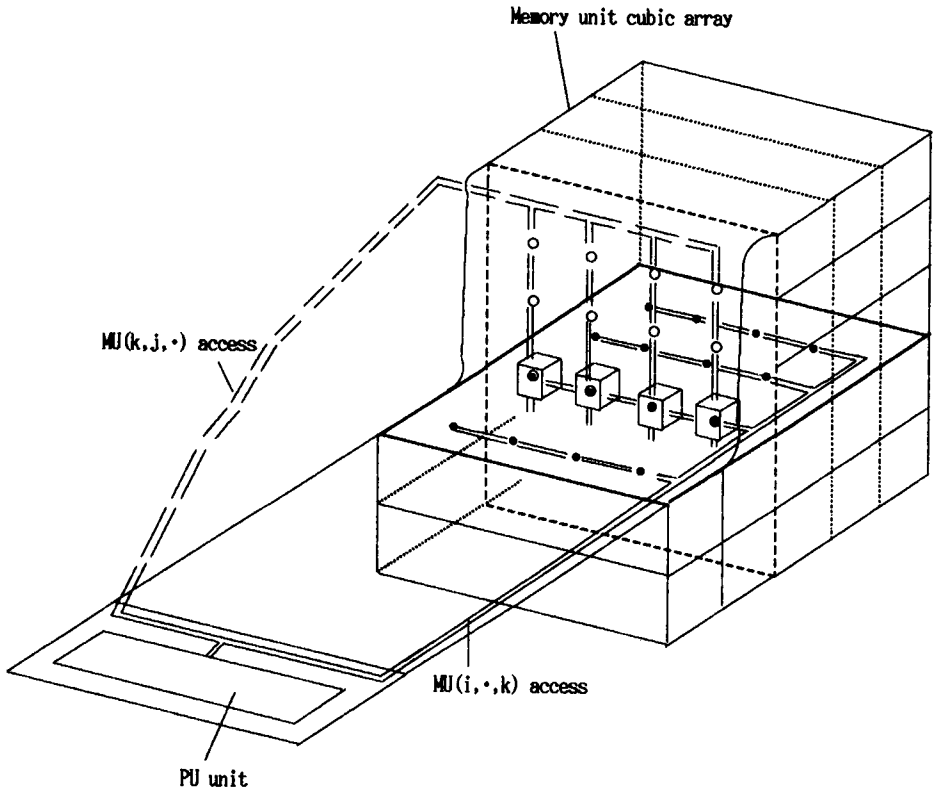


Fig. 4 Scheme of connecting a PU with memory bank unit MU's (row bus and column bus).
• ∘ : MU = : row bus === : column bus.

subarray MU(· , · ,r) is mounted on each r-th real board, shown in the figure by the square edged bold real lines (• indicates an element of MU).

Similarly, suppose that each imaginary r-th board shown by the square edged broken lines (◦ indicates again an element of MU). Both • and ◦ repesent a memory unit. In the figure, those • and ◦ on the cross line of the r-th real board and the r-th imaginary board, are contained in a small cube. Such a cube may particularly indicate that a • and a ◦ correspond to a same real memory hardware unit. On a real board, row buses are mounted (shown by double lines) and they connect those units on the board with a corresponding PU. Each set $\{MU(i, \cdot, r)\}$ for any selected i is accessed by the processor PU (r) in the so-called interleaved way. Also on an imaginary board, column buses are mounted (shown by double broken lines) and they connect those units on the board with a corresponding PU. Each set $\{MU(r, j, \cdot)\}$ for any selected j is acceseed by the processor PU(r) in the interleaved way.

We next consider an extended system which is composed of several PB's and MB's. Fig. 5 shows the system having 2×2 PB blocks, $\{PB(J,K), J,K=1,2\}$ and 2×2×2 MB blocks, $\{MB(I,J,K), I,J,K=1,2\}$. On the left (j,k)— plane, 4 PB's are designated by vertical bars. In mid-space of Fig. 5, 8 MB blocks are arranged, and each block contains N real memory boards. For simplicity, we designate only the row buses (real lines) and column buses (dotted lines) connecting memory units to those vector processors with the highest number r in any PB block. Each 'tooth' of the comb corresponds to a segment array of memory units accessed in a vector mode. Connection through row buses is shown by real lines, both inside of MB and beyond each MB. Its schema is extracted as follows:

PB(1,1)——MB(1,1,1)——MB(2,1,1)
PB(2,1)——MB(1,2,1)——MB(2,2,1)
PB(1,2)——MB(1,1,2)——MB(2,1,2)
PB(2,2)——MB(1,2,2)——MB(2,2,2)

Similarly, connection through column buses is shown by dotted lines. Its schema is again extracted as follows:

PB(1,1)——MB(1,1,1)——MB(1,2,1)
PB(2,1)——MB(1,1,2)——MB(1,2,2)
PB(1,2)——MB(2,1,1)——MB(2,2,1)
PB(2,2)——MB(2,1,2)——MB(2,2,2)

4. 2 We will give some plausible concrete designs in the following. The first example is about a case of having memory bank units of M=pN, with p=1, as shown in Fig. 6. It consists of 16 basic boards and one mother board. Every basic board constains

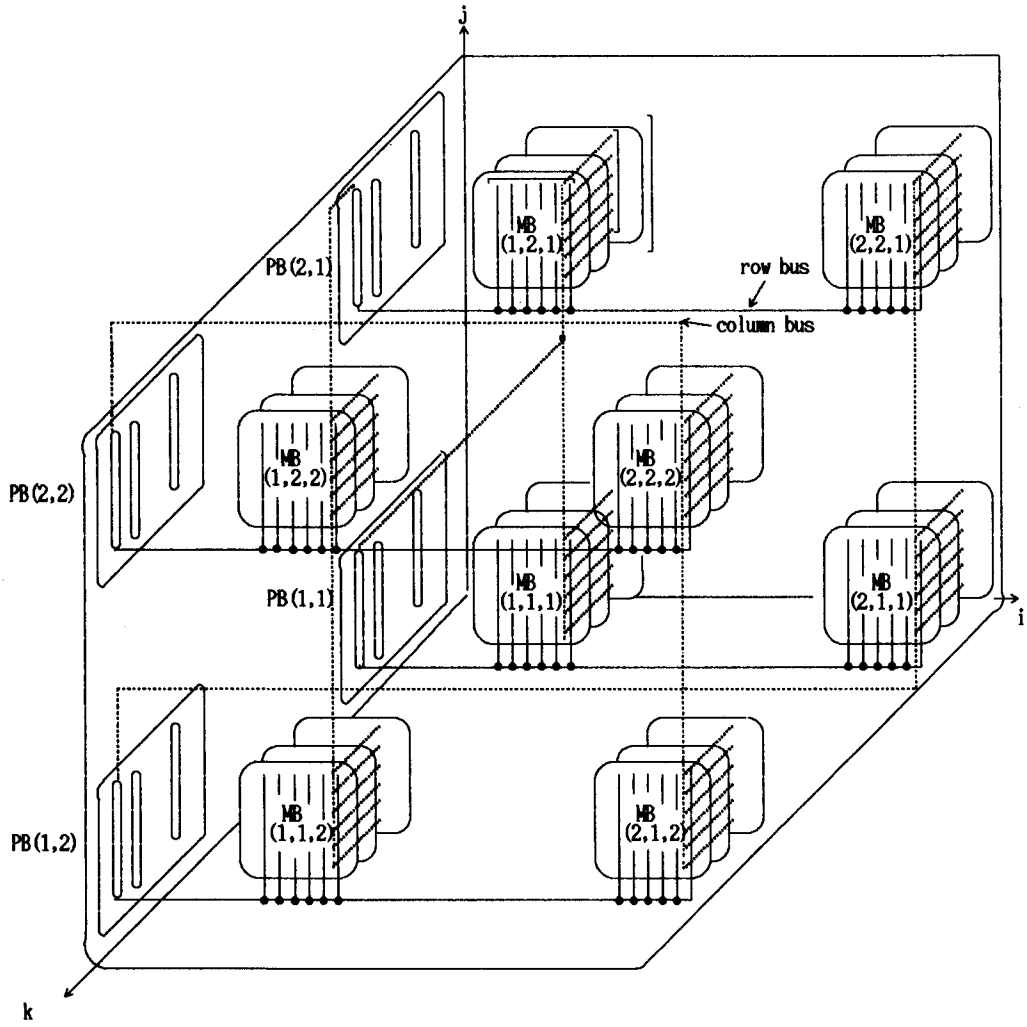


Fig. 5 (2^2, 2^3) extended system.

one vector pipeline processor unit with 16 wordlength registers (1 word =64 bits) and 16 ×16 memory units (64 Kwords per 1 unit), and hence related row buses. Each processor can have access to vectors whose components are shared by a segment array of memory units in the interleaved mode. Each basic board has 16 sets of column buses and 16 edges for communication beyond the boards. Such edges, with a total number of 256 and 16 edges of direct column buses from vector processors, are connected to the mother board on which necessary column bus network is realized. In this case, the total capacity of the main memory banks becomes 256 M words.

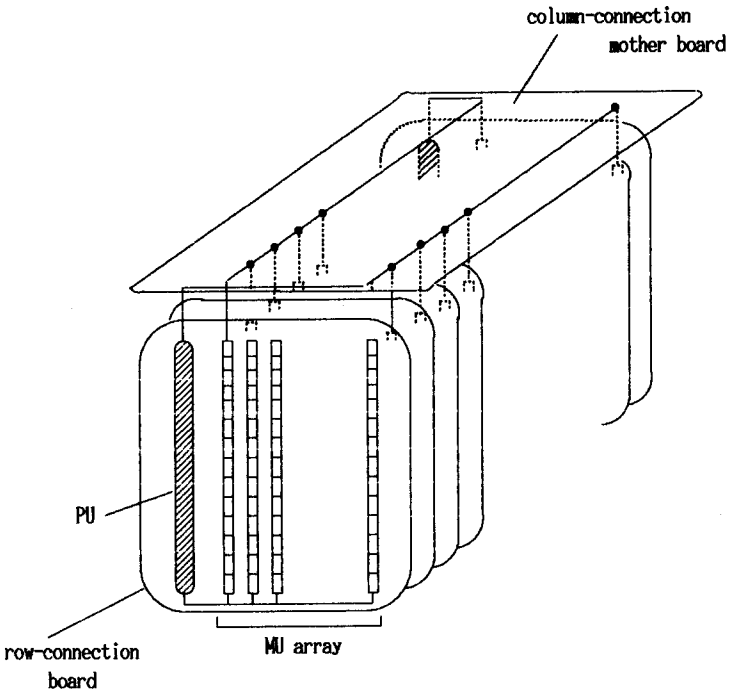


Fig. 6 Row-connection board (1 PU and 256 MU's per board) and column-connection mother board in $(1^2, 1^3)$ system.

The second example is again for $p=1$ as in the first example, but its realization is different, as seen in Fig. 7. One board constains 4 vector processing units and 16 columns of memory bank units, each column being of a 4 word length. In this board, both row and column buses are mounted and access time may be reduced.

The third example is for $p=2$ of the $(1^2,1^3)-$ system, and is seen in Fig. 8. It has the same capacity of memory as in the last example, but only two vector processing units which basically compute an 8 word length vector. In fact, an 8 word length vector is shared by two memory bank columns, each being of a 4 word length.

The final example is about the $(L^2,L^3)-$ system, and is shown by Fig. 9. It is the $(2^2,2^3)-$ system, and each PB has only one vector processor unit, and is for $p=4$. That is, the processor unit can access 4 columns (each column being of a 4 word length) in an access cycle. In Fig. 9, 'r' denotes those memory banks accessed through row buses by processors A and B in parallel, and 'c' denotes those accessed through column buses by processors A and B in parallel.

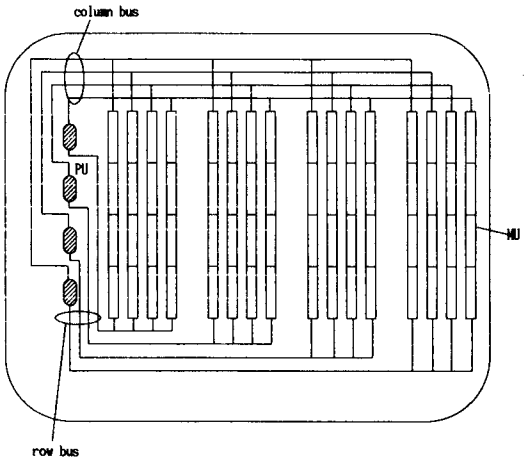


Fig. 7 Row & column connection board (4 PU's and 64 MU's per board) in (1^2, 1^3)-system.

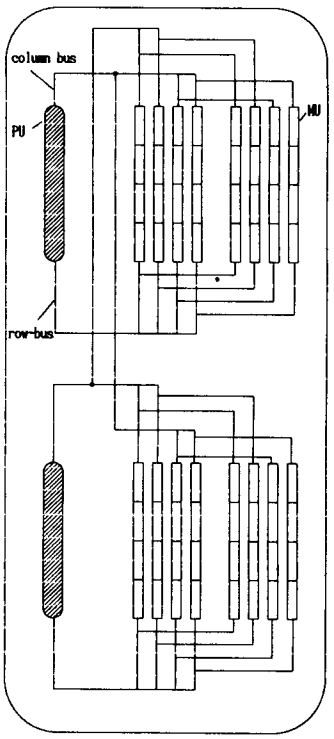


Fig. 8 (1^2, 1^3)-system with p=2.

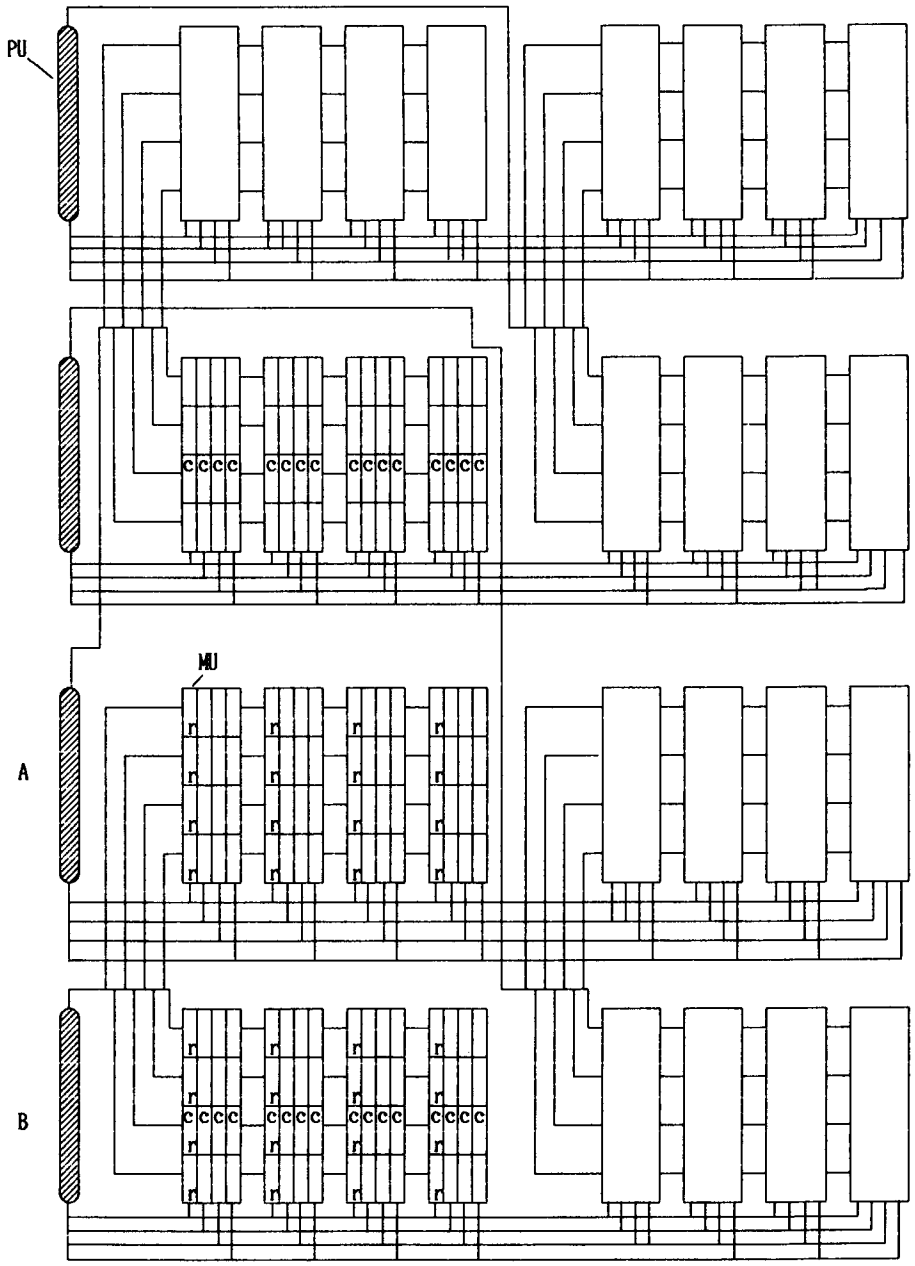


Fig. 9 $(2^2, 2^3)$-system
r : accessed via row bus from processor A and B,
c : accessed via column bus from processor A and B.

5. Performance Estimate

We will give more concrete systems that may practically be realized by up-to-date technology, and that may promise a next generation of super computers.

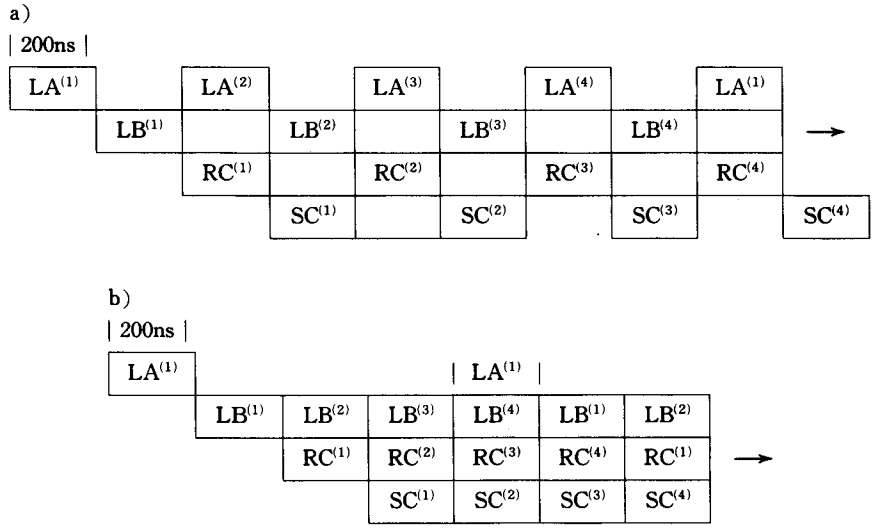
We first take the fundamental $(1^2,1^3)$— system, which is of N=16, p=4. A vector processor unit has several vector registers, each being of a 64 word length (one word is of 64 bits), and a machine clock of 3.125ns. This vector pipeline unit might execute 3.2M floating point operations per a second (3.2MFLOPS) for vectors of infinite length. On the other hand, the main memory banks part has columns (or rows) of a 64 word length, and access time of each memory unit is 200ns. Such access time is assured by today's ordinary technology, and is realized by CMOS technology. According to p=4,4 paths (each path being of 64 a bit width) are provided for 4 columns (or rows) of memory banks. In this case, the maximum bandwidth of memory access is 1.28G word.

Next, we will estimate the performance of such a single vector processor unit. For the purpose, we consider a simple arithmetic operation of two vector operands. For example, we take two variable vectors or a constant vector and a variable vector:

$$\text{a)}\quad A_i^{(p)}+B_i^{(p)} \longrightarrow C_i^{(p)} \quad (i=1,2,\ldots\ldots,64)$$

$$\text{b)}\quad A^{(p)}+B_i^{(p)} \longrightarrow C_i^{(p)} \quad (i=1,2,\ldots,64),$$

those are repeated for p=1,2,3,4,1,2,3,4,..... Denoting onetime access or operation of a 64 word length by a box [ ], we can write its computation flow as follows:



Here, 'L' means LOAD from the main memory banks to vector registers, 'R' finding

computation results, and 'S' STORE from vector registers to the main memory banks. Supposing an effective stationary state of computation sequence, we get a) 64 floating operations per 400ns (160MFLOPS), b) 64 floating operations per 200ns (320MFLOPS).

Summing-up the above estimate for $N=16$ vector processors, we get a total performance a) 2.56GFLOPS, b) 5.12GFLOPS.

Using the system mentioned above as a unit, we construct the $(2^2,2^3)$— system. Then we get 4 times performance of a single unit's, a) 10.24GFLOPS, b) 20.48GFLOPS.

These estimates give promise for our vector parallel computer to be a candidate for the next generation of super computers.

Acknowledgement

The autor thanks all the staff of Matsushita Electric Industrial Company engaging in the development of ADENA-II for their encouragements, and also to Miss N. Maruyama for her typewriting and tracing the author's manuscripts.

References

1) R. M. Hord, The ILLIAC-IV the First Supercomputer, Computer Science Press, Maryland 1982.
2) S. F. Reddaway, DAP-a distributed array processor, lst Annual Symp. on Comput. Architecture (IEEE / ACM), Florida, 1973.
3) T. Hoshino, T. Shirakawa, T. Kamimura, T. Kageyama, K. Takenouchi, H. Abe, S. Sekiguchi, Y. Oyanagi, T. Kawai : Highly Parallel Processor Array "PAX" for Wide Scientific Applications. 1983 Intern. Conf. on Parallel Processing, IEEE, pp. 95-105, 1983.
4) T. Nogi, and M. Kubo, ADINA Computer I, I. Architecture and Theoretical Estimates, Memoirs of the Faculty of Engineering, Kyoto University, 42 (4), pp. 421-439, 1980.
5) T. Nogi, ADINA computer II, I. Architecture and Theoretical Estimates, ibid., 43(1), pp. 124-144, 1981.
6) T. Nogi, ADINA Computer I and II, II. —Data Structure, ibid., 43(3), pp. 434-450, 1981.
7) T. Nogi, Parallel Machine ADINA, in Computing Methods in Applied Sciences and Engineering, V, eds. R. Glowinsky and J. Lions, North-Holland, pp. 103-122, 1982.
8) T. Nogi, The ADENA Computer, in International Symposium on Applied Mathematics and Information Science, Kyoto University towards Multidimensional Flow Models, Mathematics and Computers, pp. 7/9-16, 1984.
9) T. Nogi, Parallel Computation, in Studies in Mathematics and its Applications 18, Pattern and Waves, Kinokuniya/North-Holland pp. 279-318, 1986.