Indirect memory decoding for vector access

J-L. DEKEYSER*
P. PREUX*

* *LIFL - USTLFA Cité scientifique 59655 Villeneuve D'ascq Cedex FRANCE*

Abstract

An attached array-processor architecture, called WEST co-rocessor, is proposed in this paper. This WEST architecture is omposed of the following units : a control module for memory ccess which allows indirect memory access for vector element gather / scatter), multi-SIMD processing unit including several ector processors performing vector functions over same or different ectors, one or more scalar processors for reduction operation rocessing, a multi-access memory ensuring 16 simultaneous nputs or outputs. The memory address management unit is ompletely specified and some examples of tasks achievement are emonstrated. This new approach of array-processor should allow o execute efficiently vector algorithms referencing randomly istributed data instead of traditional scientific algorithms as natrices and array manipulation.

Introduction

There are two types of vector processing : vector pipeline nd array processor.

The concept of pipeline processing is similar to assembly nes in a factory. The pipeline consists of a cascade of processing tages. Each stage performs arithmetic or logic operations over a ata stream crossing the pipeline. The final result is produced by ie last stage. This approach is used in the most recent upercomputers : CRAY, ETA 10, IBM 3090/vf600...

An array processor consists of multiple processing ements (PEs). An array processor can handle Single Instruction nd Multiple Data streams (SIMD). It is specially designed to erform vector computations of matrices or arrays of data. The ost popular array processors are : ILLIAC IV, BSP, Pepe, FPS...

The WEST project consists to design a user-friendly nvironment for work-stations in vector programming context. It nsists of several parts : vector language [1] [2], debugging tools, mulation tools, ... One part of this project is the development of tached array-processor.

Vector algorithms evolution [3] obliged programmers to e more and more sparse vector operations. Index vectors control e access to the sparse vectors : gather / scatter. WEST aims to ke account of these indirect memory accesses by the memory ldress management unit (MAMU). Effective addresses are ocessed by this MAMU independently of processors. A multi-cess memory (MAM), that provides simultaneously 16 inputs or outputs, ensures a constant data stream by deleting conflicting cess to the same memory bank. Once the global WEST chitecture has been defined, a detailed simulation of each nctional units is set in motion.

After a general presentation of the attached array-ocessor, we will present a detailed account of the MAMU. nally, some examples of communication protocols will be monstrated.

1. The array-processor architecture

1.1. Introduction

The WEST computer comprises the following units (fig 1) :

. the sequencer : it controls data transfers between the fferent units. It also dispatches the processing of vector functions er all the active units to satisfy the host computer requests.

. vector processors (VP) : each one consists of sixteen ocessor elements (PE) executing similar vector functions nultaneously over sixteen different sets of data.

. the scalar processor (SP) : it produces a scalar result from sixteen data onwards. It is used to execute vector reduction functions.

. the multi-access memory (MAM) : it can execute sixteen inputs or outputs simultaneously for any sixteen addresses.

. the memory addresses management unit (MAMU) : it ensures the transfer of data between the different units, controlled by the sequencer by using a multi-bus which allows transfers of sixteen addresses and sixteen data simultaneously. It effects locally the indirect access necessary to process gather and scatter. The VP and SP share the MAM in turn, the sequencer centralizing the control.

1.2. The sequencer

The sequencer responds to two different types of activities. On the one hand, it controls the MAMU to manage communications without producing conflict between the units. Several control signals select the type of data, input or output, the direct or indirect access to the memory and the type of communication with MAMU and the different units. Synchronization lines trigger the transfers and ensure that the end of the transfer in taken into account. Supplementary lines specify the length of the vectors, addresses of data vector and index vector. On the other hand, it triggers the VPs and the SP. It controls these units and tells them which task to perform and here too synchronization lines trigger and ensure that the end of operation is taken into account. A sequencer specific memory contains global vector functions which must be carried out to satisfy the requests of the host machine.

1.3. The vector processors (VP)

Each of the VP consists of sixteen processing elements (fig 2). They are connected to the MAMU via a multi-bus of data (MBD - sixteen buses of 32 bits). Each bus is linked to the PE bus permanently. The PE work autonomously and asynchronisticly. Controlled by the sequencer, they load and unload the data into their bus and perform specific tasks contained in ROMs and RAMs. Once finished, each PE sends a synchronization signal. The sixteen signals (one signal per PE from the same VP) are expected in order to send to the sequencer one single signal which signifies the end of operation. Each PE register forms a sixteen length vector register at the VP level. The set of vector instructions can use explicitly these vector registers.

The use of several VPs permits two types of functions :

. SIMD : all the VPs carry out the same vector function on slices of sixteen different data. The VPs are de-synchronized to allow multiplexing of data in the multi-bus.

. MSIMD (Multi-SIMD) : each VP carries out a different vector function. This technic is very efficient for vectors which are shorter than sixteen..

1.4 . The scalar processor (SP)

The SP is connected to the MBD. It performs reduction vector operations on sixteen different elements of the same vector. As with PEs, it has a RAM and ROM which ensure the storage of micro-functions, as well as certain numbers of vector and scalar registers directly accessible by the set of vector instructions. It takes its loading, unloading and task trigger from the sequencer. The result produced will be placed in one of the sixteen MBD buses. Synchronization mechanisms allow communication between the SP and the sequencer.

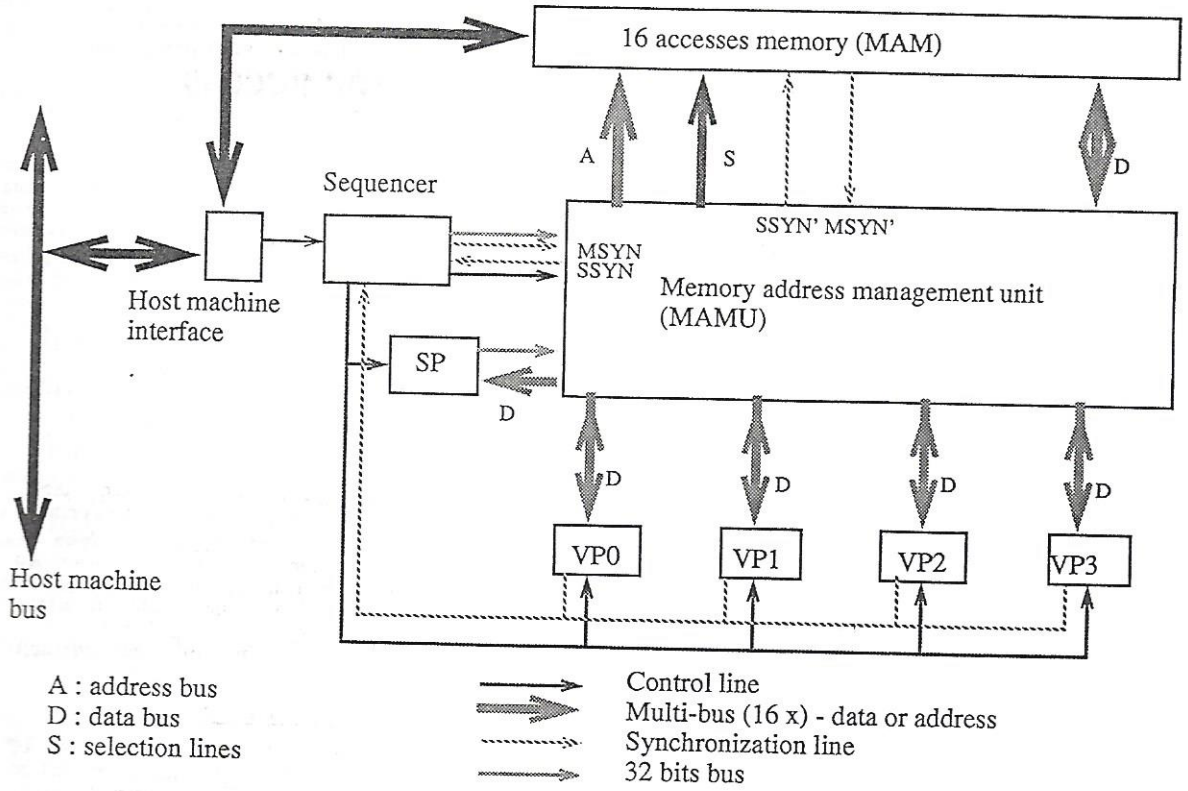

Fig 1 : The WEST architecture with 4 VPs

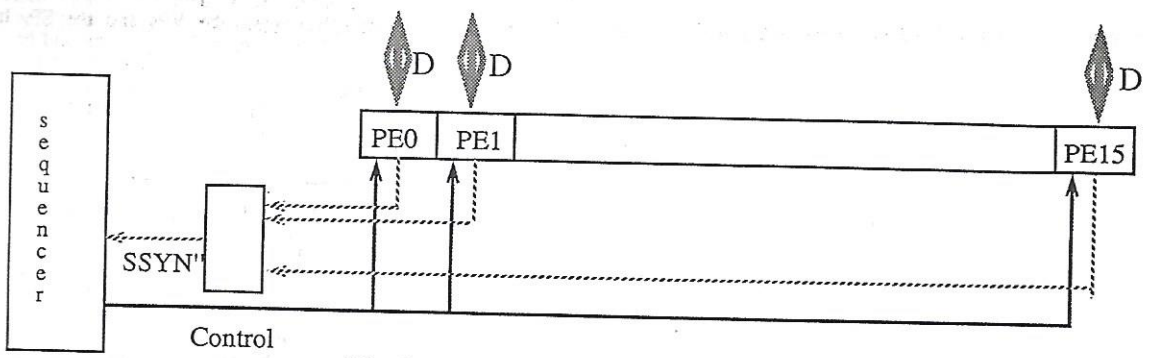

Fig. 2 : Internal structure of a VP

1.5. The 16 accesses memory (MAM)

This memory possesses two essential characteristics :

. it authorizes sixteen simultaneous accesses to any sixteen words;

. the accesses are either all inputs or all outputs. The PEs of anyone VP possesses the same output or input instruction (STORE, LOAD).

The MAM communicates with the MAMU in asynchronic mode.

Different approaches are envisaged :

◆ A MAM simulation uses classic memory technologies and technics, but this does not ensure effective simultaneous access in case of conflict. Various technics for this simulation are possible :

the use of interleaved memory banks connected to an alignment network [4];

the use of a multi-plans memory composed of sixteen identical memory plans each containing a copy of the memory. Each plan is composed of sixteen interleaved banks. Sixteen simultaneous inputs are executed in sixteen different plans : simultaneous multiple access is therefore guaranteed in the input independently of the addresses produced. In the case of sixteen outputs, two different possibilities

- in the case of one output based on sixteen contiguous addresses, the sixteen outputs are executed simultaneously in the sixteen banks. This is the case for each of the sixteen plans keeping coherence between the plans.

- if the sixteen addresses are randomly distributed, the sixteen outputs are temporary stored in sixteen FIFOs (one per bank) and the outputs are realized simultaneously for all the addresses at the top of the list until the sixteen FIFOs are empty.

The asynchronic character of the MAMU / MAM communications is advantageous in the definition of this memory.

◆ An elementary memory point offering sixteen parallel accesses in either input or output suppresses the access conflict. Sixteen address decoders in parallel select the same memory point. This memory point possesses an access time of the same order as that of traditional memories. Its integration complexity is nevertheless more advanced. Studies of function of the memory in

VSI are being carried out at the moment. On the other hand, ew approaches in 3-dimensional integrated circuits seem very ttractive. The MAM interface is composed of sixteen bi-irectional buses of data, sixteen address buses and sixteen input election lines; each of which is linked to a link bus : data bus / ddress bus. The length of the vector which is to be processed etermines the number of the first lines to be selected.

This interface also includes a read / write line and two ynchronization lines connected to the MAMU. When the election lines and the buses contain data, the MSYN is released. When the memory access, the MAM can release the SSYN.

Finally, the host computer integrates the MAM in its emory space by a direct access via its own bus. A specific nterface connects the host bus to one among sixteen buses of the ulti-bus. A supplementary line selects the communication type : AM / host or MAM / MAMU.

1.6. The memory addresses management unit (MAMU)

The MAMU function is to organized the data stream etween memory and processors. It executes in a vector way the direct access to the memory without external intervention. ransfers MAM to MAM are exclusively processed by the MAMU. he sequencer sends a request to the MAMU, specifies the ddresses, the type of data and the type of access. ynchronization lines allow communication between sequencer nd MAMU and between MAM and MAMU.

2. Specification of the memory address management unit (MAMU)

After specifying the meaning of signals in input and utput, we will present a detailed account of the three main parts of this unit (fig 3) :

- control unit
- address production unit
- address / data exchange unit

2.1. MAMU's external connections

The MAMU is connected to all the active units of WEST : the sequencer, the MAM, the VPs and the SP. The junctions between MAMU and sequencer consists of :

. vector length LG on a 4-bits bus

. one $\overline{r/w}$ line

. one $\overline{f/d}$ line denoting the data format (simple or double precision)

. one $\overline{d/i}$ denoting the addressing mode (direct or indirect)

. one $\overline{l/p}$ denoting the transfer class (between processor to memory or memory to memory)

. 2 address buses IND and V

. 2 synchronization lines sequencer / UGAM :

MSYN : releasing activities from the sequencer

SSYN : completion of activity from the MAMU

The 4 lines $\overline{r/w}$, $\overline{f/d}$, $\overline{d/i}$, $\overline{l/p}$ define the task to execute by the MAMU. Table 1 indicates the different tasks to perform according to the values of these four input lines.

The connections with the VPs just consist of 16 bi-directional data buses. The sequencer synchronize the exchanges between VP and MAMU via LOAD / STORE vector instructions.

Connections with the SP are made of 16 uni-directional

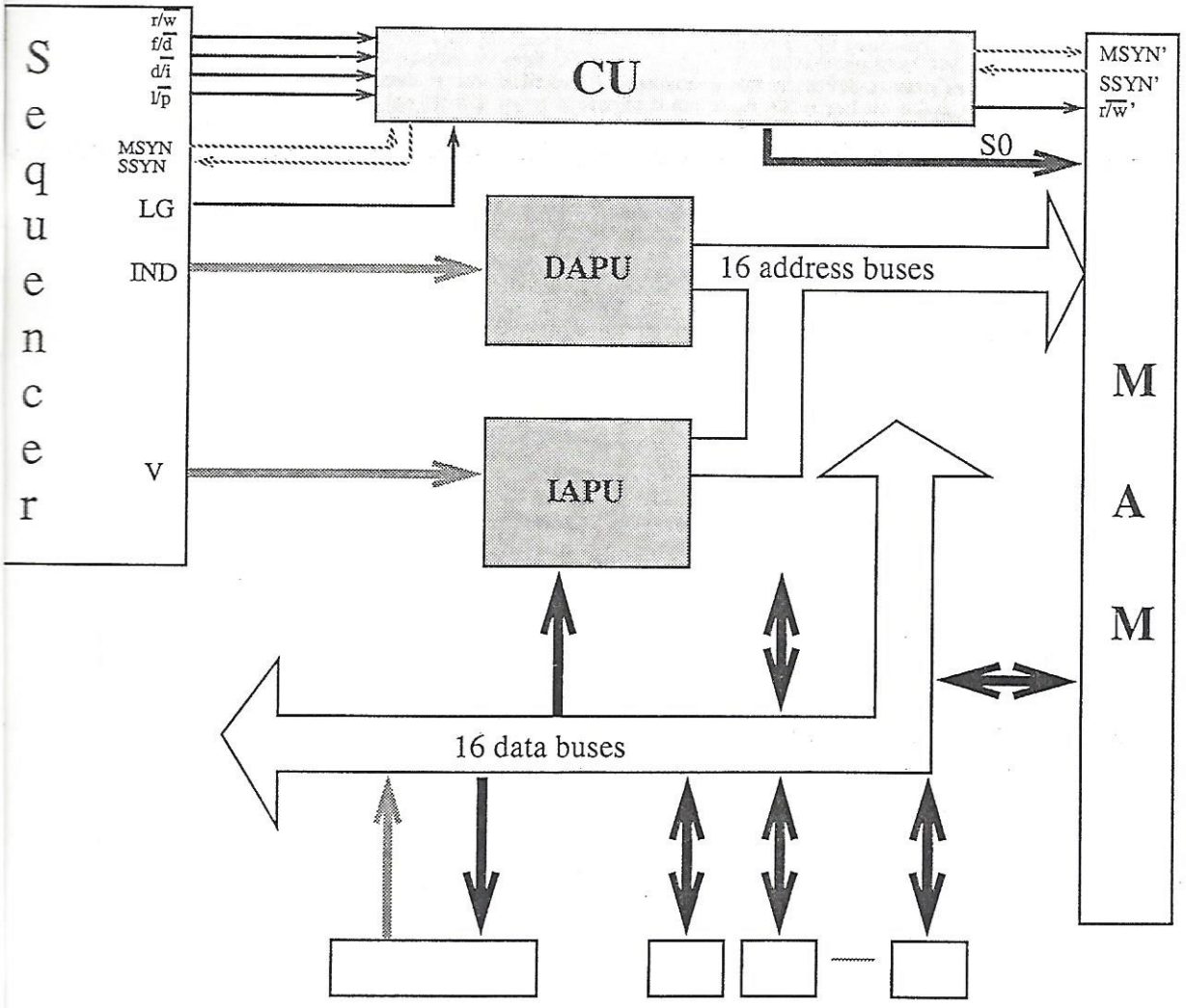

Fig 3 : General structure of the MAMU

data buses to load the data in the SP, and one uni-directional data bus to store the result.

The connections to MAM have been described in section 1.5.

2.2. The control unit (CU)

This unit controls the whole MAMU's activities. CU receives the requests from the sequencer, emits requests to the MAM and manages the data and the addresses streams inside the MAMU. Its action is totally fixed by the four inputs $\overline{r/w}$, $\overline{f/d}$, $\overline{d/i}$ and $\overline{l/p}$ when MSYN is validated by the sequencer. The parameters of the memory access are memorized from IND, V and LG buses. LG allows to specify which buses carry valid informations.

2.3. The address production unit (APU)

The address production unit computes the effective addresses for vector accesses. It is divided in two parts . On the one hand, the direct address production unit (DAPU) is used for all memory accesses. On the other hand, the indirect address production unit (IAPU) is only used when an indirect mode access is performed.

DAPU and IAPU are mainly composed of adders.

There are 4 different kinds of effective address computing :

◆ accessing A, a simple precision vector

Let α be the base address of A, which is read on IND bus. By incrementation of α, the DAPU computes the effective addresses :

$$A(0) \equiv \alpha^{1)}, A(1) \equiv \alpha + 1, \ldots A(LG-1) \equiv \alpha + LG - 1$$

These LG addresses are placed on the multi-bus of address. The access to the MMA can be performed.

◆ accessing A, a double precision vector

A double precision data is stored in two contiguous memory words. The access to this kind of vector is composed of two phases : less significant words access followed by the most significant words access. The DAPU computes the addresses of the less significant words. A first memory access is performed while the DAPU computes the addresses of the most significant words. When the first access is completed, a second access is performed.

◆ accessing indirectly A, a simple precision vector, using I as an index vector

$\overline{f/d}$	$\overline{l/p}$	$\overline{d/i}$	$\overline{r/w}$	Operation
0	0	0	1 ..	external scatter of doubles : AD (I) ⟸ MBD
0	0	0	1 ..	external gather of doubles : MBD ⟸ AD (I)
0	0	1	0 ..	external writing of doubles : AD ⟸ MBD
0	0	1	1 ..	external reading of doubles : MBD ⟸ AD
0	1	0	0 ..	internal scatter of doubles : AD (I) = DD
0	1	0	1 ..	internal gather of doubles : AD = DD (I)
0	1	1	x ..	transfer of doubles : AD = DD
1	0	0	0 ..	external scatter of simples : A (I) ⟸ MBD
1	0	0	1 ..	external gather of simples : MBD ⟸ A (I)
1	0	1	0 ..	external writing of simples : A ⟸ MBD
1	0	1	1 ..	external reading of simples : MBD ⟸ A
1	1	0	0 ..	internal scatter of simples : A (I) = D
1	1	0	1 ..	internal gather of simples : A = D (I)
1	1	1	x ..	transfer of simples : A = D

In this table, A and D are simple precision vectors, AD and DD double precision vectors, I is an index vector and MBD the multi-bus of data

Table 1 : Different operations realized by the MAMU

1- this means that the effective address of A(o) is a

Let α the base address of A be read on bus V, and γ the base address of I, be read on the bus IND. The DAPU computes the effective addresses of :

$$I(0) \equiv \gamma, I(1) \equiv \gamma + 1, \ldots I(LG-1) \equiv \gamma + LG - 1.$$

A reading access to the MAM is performed to load the effective value of the indexes into the MAMU. Then, the IAPU computes the effective addresses of the elements :

$$A(I(0)) \equiv \alpha + I(0), A(I(1)) \equiv \alpha + I(1), \ldots,$$

$$A(I(LG-1)) \equiv \alpha + I(LG-1).$$

A second access to the MAM is performed.

◆ accessing indirectly A, a simple precision vector, using I as an index vector

This operation is also decomposed in two parts : less significant words and most significant words accesses. An access to the MAM performs the reading of the index vector elements which are directed to the IAPU. This one computes the effective addresses of the less significant words. Then, these words are accessed. The IAPU computes the effective addresses of the most significant words and then access them.

2.4. The address / data exchange unit

This unit is composed of a multi-bus of data (MBD) and a set of 16 latches (TMP) eventually achieving a temporary storage of vectorial data. These latches must be used to perform internal operations in the MAMU ($\overline{l/p}$ = 0) in indirect addressing mode ($\overline{d/i}$ = 0) - to realize internal gather / scatter (A = B(I) or A(I) = B).

3. Communication protocols study

Now, we demonstrate the communication protocol between the sequencer, the MAMU, the MAM and the processors, for three operations. The actions realized by each unit are also described. These examples are :

- A = B : internal assignment of B in A with LG <= 16 (fig 4)
- A = B (I) : internal assignment of the gather of B under the control of the index vector I with LG <= 16 (fig 5)
- C = A + B : addition of A and B by the PVs, the result being stored in C (fig 6)

Conclusion

The strict definition and studies of the other units of WEST are in progress. Simulations, using the LIDO system [5], will allow to validate their mechanisms. Following these former concepts, a few remarks can be achieved :

a) traditional interleaved memory banks do not permit an effective multi-access to the same bank : a temporary storage mechanism is required . At the opposite, the MAM put the access conflict back to the memory word level. A hardware mechanism taking account of writing priorities removes the sequential accesses caused by conflicts. This kind of memory always replies in a constant time whatever the generated addresses are.

b) the integration complexity of MAM comes up against the actual technological possibilities. A simulation of this kind of memory using multi interleaved memory banks eliminates most of the cases of conflicts and offers performances close to the optimal solution using actual technologies.

c) the complexity concerning the connections between the MAMU and the different units (approximately 3500 wires for 4 VPs of 16 PEs each) seems to be the most awkward point of this solution. To solve this problem, a few ways have to be considered (such as multiplexing optical fibers) and then the wires number can be reduced by decreasing the number of PEs per VP. Simulations will allow us to evaluate the performances which depends on the number of PEs and the number of VPs.

d) the sequencer has a key-role in the execution of vectorial functions. Its load depends on the number of units. To avoid an overload risk, the spread of the vectorial functions over the processors is handled at compile-time by the host machine

e) the performances of WEST mainly depend of the functions to execute. As the VPs have their own registers, the more CPU time-consuming the vector functions are, the less the MBD is loaded. This will have to be taken into account by the vectorial machine language which includes a lot of advanced vector functions (sin, cos, exp, log, ...) and possibility for the user to create its own functions.

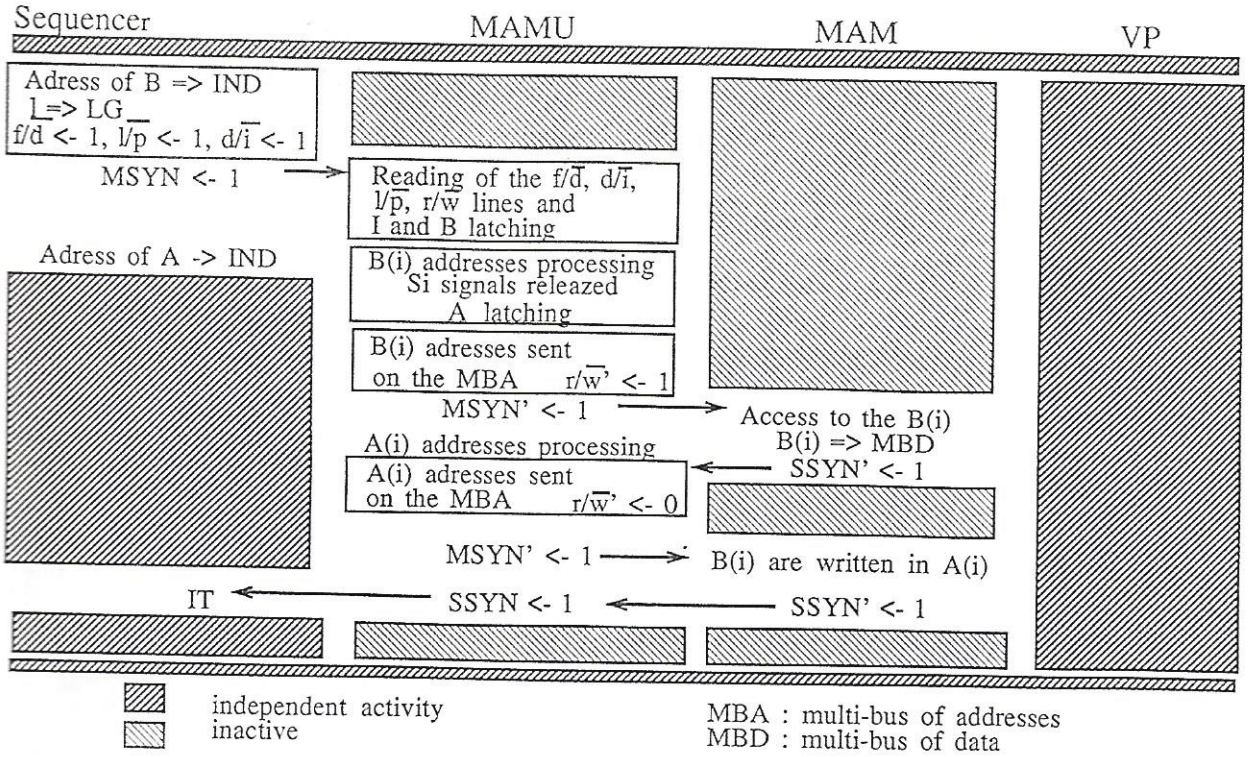

Fig 4 : Communications protocol for the operation A = B

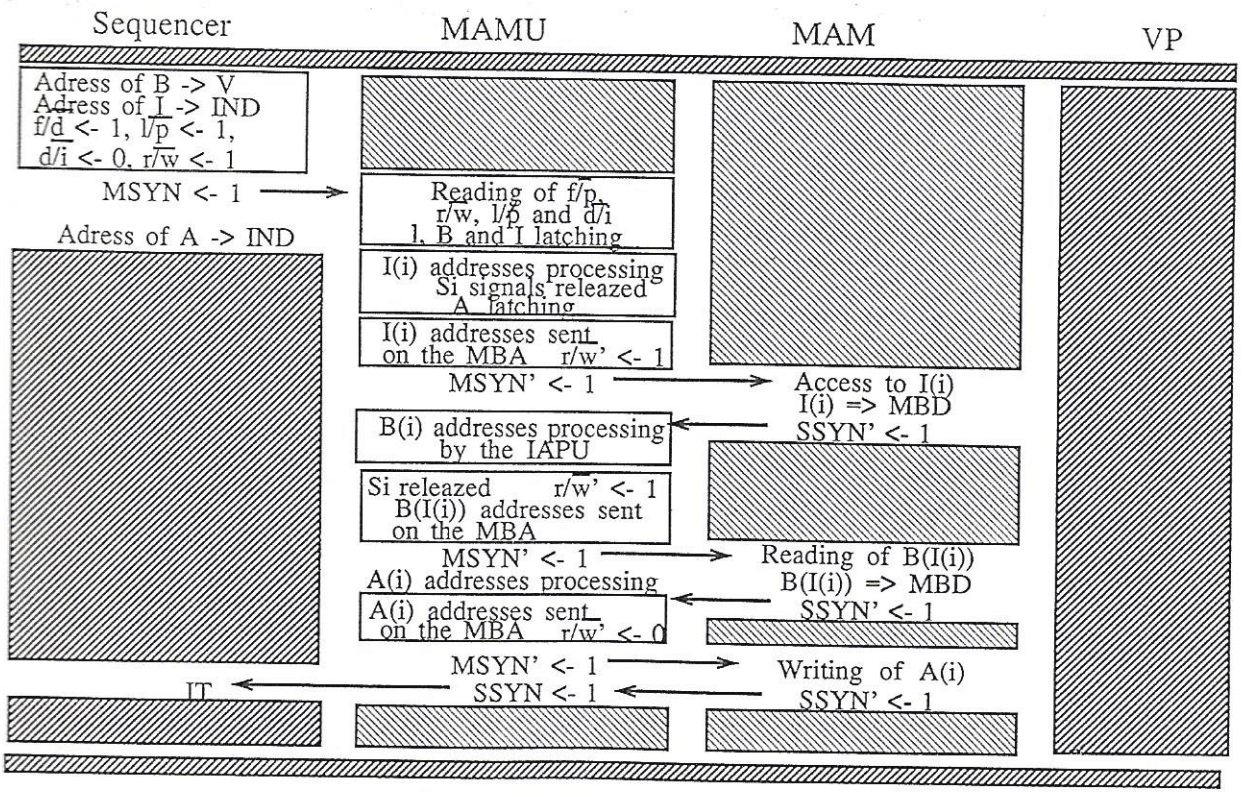

Fig 5 : Communications protocol for the operation A = B (I)

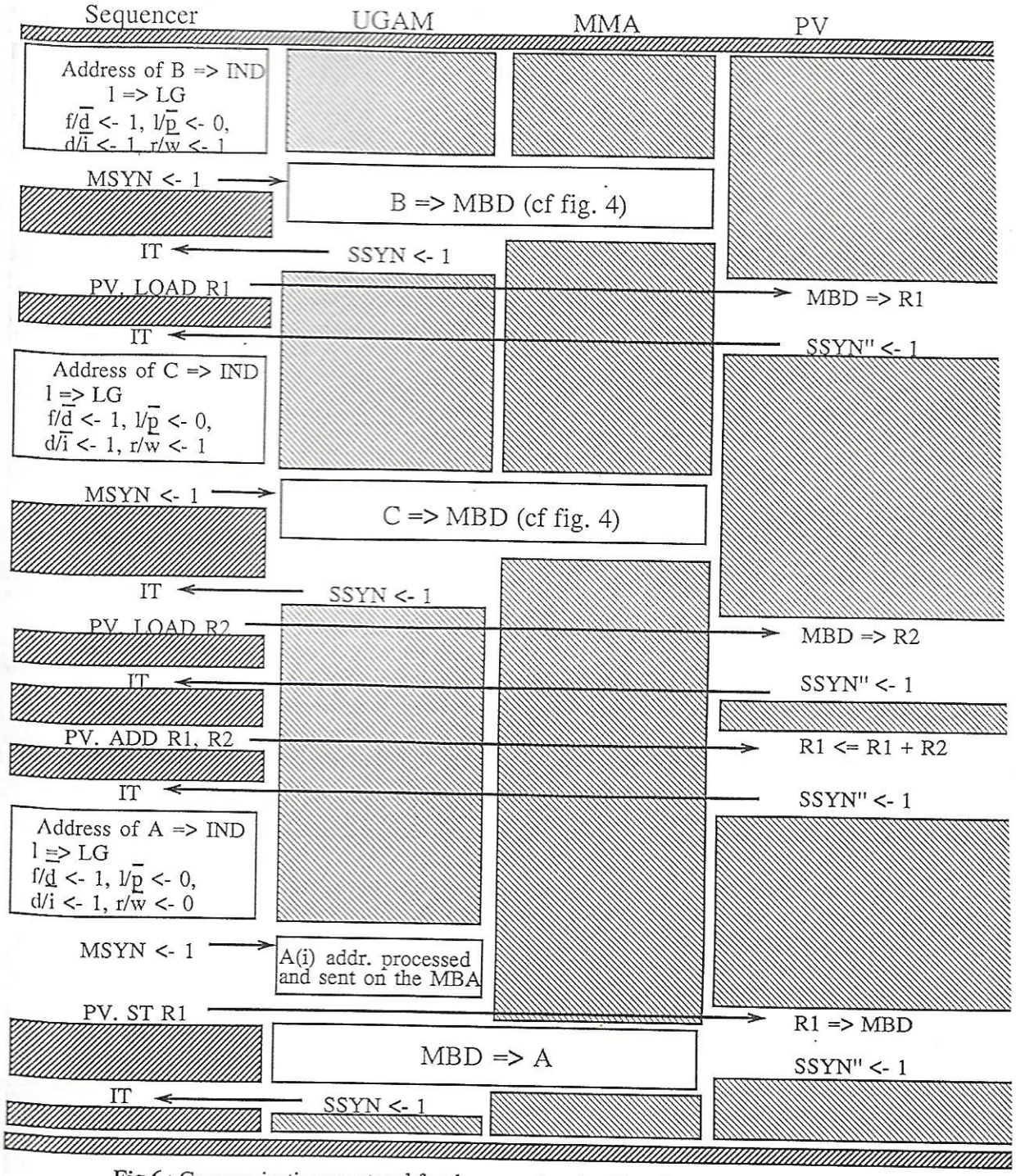

Fig 6 : Communications protocol for the operation A = B + C

Bibliography

[1] J-L. DEKEYSER, F. HANNEDOUCHE, G. RICCARDI, J. VAGI, "The AFTRAN vector preprocessor project", FSU-SCRI-88-43 Tallahassee Florida

[2] V CHANDRA, J-L DEKEYSER, F HANNEDOUCHE, G RICCARDI, J VAGI, "AFTRAN : array FORTRAN programming language", FSU-SCRI-89T-16

[3] J-L. DEKEYSER, "Vectorization of the GEANT3 geometrical routines for a Cyber 205", Nuclear Instruments and Methods in Physics Research A264(1988) 291-296

"Technical report on the vectorization of GEANT3", Proceedings of the workshop on SSC simulation Argonne Illinois August 1987

[4] P. PREUX, "Conceptual studies of an inter connection network for array-processor", Mémoire de DEA - LIFL USTLFA Villeneuve D'Ascq

[5] P. BAKOWSKI, "LIDO - A Silicon Compile Preprocessor", Microprocessing and Microporgramming - 1987 - number 20, p 167-172