The "farm"approach to the second level triggering for future HEP detectors

Vasile Buzuloiu
Institutul Politehnic Bucureşti and CERN/LAA project

1. Introduction.

1.1. The assumption about the data stream.

In the following analysis it is supposed, as usually speaking about the second level triggering [1], that the amount of the data from detector(s) - of the order of 1 Mbyte per bunch crossing - is reduced by the first level triggering to a rate of 10μs and to a volume of about 2 kbytes i.e. only so called regions of interest pointed by the first level triggering machine are further sent to the second analyzing /deciding machine. These two parameters, combined, give the main characteristics of the input : 200 Mbytes/s which is at the limit of what the systems under development for other applications handle today; still we must be aware of the fact that the supposed reduction factor of about $5*10^5$ by the first level triggering - from 1 Mbyte every 15 ns to 2 kbytes every 10 μs -could be smaller and these 200Mbytes/s must be thought as a minimum.

1.2. General remarks on data (signal) processing functional bricks.

Our signals are complex ones (the events are very complex), their appearance intuitively leads to characterize them as images and what we look for are patterns which we have to distinguish by their features (themselves difficult to choose - to define). The general philosophy, borrowed from image processing techniques, is to concentrate a feature in a point by applying transformations to the input data bulk. Usually these transformations are split up in sequences of simpler ones : linear integral transformations and point nonlinear transformations. For the last ones the nowaday technology allows a very fast implementation when the values are in a reasonably (not very large) range, namely the "look-up table" (LUT) techniques. For the former ones the core is the "dot product" operation : an expression of the form $\sum x_i y_i$, which is used for convolution.

The need of fast multiplying has led to supplementing the processor's ALUs with (integer) multipliers or (floating point) coprocessors. In this respect the image processing architectures went further : There are true convolution units (convolvers) for standard TV real-time which accept nuclei up to 15x15.

The algorithms tested so far for future detectors data (mostly simulations) mainly enter in the class just described. Hence the advantage of image processing systems in building second level triggering machine. Nevertheless the commercial image processing systems do not have neither the speed nor the flexibility we need in the second level triggering machine because they are intended for standard image processing tasks. Concerning the flexibility they give precedence to modern microprocessors.

1.3. About possible structures of the machine.

It was recognized the basic distinction between decision rate imposed to the machine (10 μs) and latency (the period needed to process the bulk of event data from the input instant to the decision or, equivalently, the run period of the trigger algorithm)[2]. The later can exceed the former by a large factor and, in fact, this is the only way to reach the necessary decision rate: to allow enough latency. This means either a single stream of data jumping serially into the cells of a long chain of pipelined processors, or a multitude of parallel streams passing through one processor which accomplishes the entire processing job in a longer-than-decision period.

These two solutions can be implemented in different ways :

a) A dedicated architecture (fully custom designed) optimized for a class of algorithms.
b) A commercially available general purpose massively parallel supercomputer.
c) Commercially available pipelined image processing systems.
d) Semi-custom designed systems.

We shall put the "farm" in the last category for we envisage not simply a farm of workstations (what was called "brute force"); we think to a farm of processors adapted to the peculiarities of our task and built around a powerful microprocessor.

The first way can produce the fastest machine for precisely defined problems, though it is the most rigid. The second one is very flexible, very expensive, and not fast enough (not yet).

The third could manage the data stream also as a small farm of such systems working in parallel; still the inherent restrictions in flexibility are the same. In all cases the need for interfacing the second level triggering machine with the first level triggering one is the custom job. Multiple HIPPI interfaces will be used [2].

1.4. The comparison criteria to be taken into account.

As pointed out in [2], the technical performances to be considered are : the decision rate and latency for benchmarking algorithms or equivalent ones; also degree of parallelism and complexity of the machine, and flexibility (the ability to be adapted to other algorithmic tasks and invariance to the input data format and output data format). There are also factors like the estimated developing time, and the cost - prototype and machine - which will decide the advantage of a given solution with respect to others.

2. The Farm.

2.1. Why a farm can be the right choice.

One of the main assumptions about the data made so far is that they come ordered in time to the triggering machine, i.e. all the data pertaining to a bunch crossing are in the same stream (this time-ordering could not be trivial in machines 20 m long and a few meters in a diameter if we take into account that the light speed is only 30cm/ns and the bunch crossing rate is 16 ns; but this is a task of the front end electronics). So, the successive "images" are uncorrelated and *every set of data can be and will be processed independently*.

This fact, implicitely accepted in any approach so far, is essential for the opportuneness to use just a "farm". We have to build the processors so that each has enough computing power to accomplish the job for an event, or only for a region of interest, within a reasonable time. That is possible with a well balanced structure which contains less than a full image processing system and at the same time is enough flexible.

2.2. The block diagram.

Fig. 1 depicts a block diagram of a farm fed through a HIPPI interface. Each block of data, corresponding to an event is fed into (the memory of) a processor. The next block is fed into another processor. Consequently a commutator is needed at the input of the machine and a controller for it, so that always the new data block reaches a free processor. One must only make provision of enough computing power (number of processors) as not to overload the machine when biggest blocks would come one after another.

The data blocks are not necessarily equal because the regions of interest of successive events can vary in number and size; so, the computing time for different events will differ and the machine will work in a macro-asynchronous regime. It is the controller of the commutator which distributes the successive jobs to the processors, having as its input the information about the working state from all processors.

Whatever algorithm machine will run, we know that the decision about an event is short i.e. the result of processing the 2 kbytes or more will be contained in a few bytes. So the output of any processor does not put any hard problem of communication compared with the input. As the machine has an unique output for the external world, a global decision processor (or a decision manager) must collect every particular decision from farm members and further process them. Apart from a possible processing at a higher level (comparison of region of interest decisions and the dialogue with the external world) the task of this last processor is to re-order the flow of decisions due to the fact that having various processing times results in changing the time-order of the decisions.

Finally, the processors are to be defined. It is the peculiarity of signal processing algorithms which must be considered. The benchmarking algorithms used so far show once more that the main macro-operation is the dot product and the gain in speed depends essentially on a fast implementation of it.

2.3. The input commutator and its controller.

Suppose a processor has just finished its current task. The actions to be done by it are: send the decision to the global decision processor and inform the controller of the commutator that it is ready for the next task. The controller can be simply a FIFO initially containing the processor numbers in natural order. Every 10 μs a step forward is made. When a processor finishes a task it sends its number as a new input to the FIFO. If the tasks are too time consuming, or the number of processors is too small, the FIFO will become empty and the new input data will be lost; that must be acknowledged. But this situation has to be avoided from designing phase. As the machine is fully scalable, more processors must be supplied to the farm. The commutator is controlled straightforward by the output of the FIFO (Fig. 2).

2.4. The re-ordering of the decisions.

The result of the event data processing sent to the global decision processor contains also the labels of data (the event number, regions of interest position). According to the event number, the results will be stored in the processor memory, naturally ordered. This allows the simplest recovery of the order (Fig. 3). The global decision processor outputs a decision every 10 μs. On the average, its inputs (coming from the processors of the farm) are of the same small volume as the outputs are, with a rate of one every 10 μs - or at most a few - (depending of the distribution of tasks: one region of interest per processor or one full event per processor). This allows enough time for supplementary computation at higher level (above the level of the farm processor task). The supplementary computation might be the comparison of the results for various regions of interest of the same event.

2.5. The processors.

We envisage the use of modern fast RISC microprocessors or DSPs as the core of the processor. It means a 32 bit engine which executes most instructions in a single machine cycle and whose integer arithmetic precision is sufficient for most needs of our signal processing tasks. It must also be a chip for which the developing tools are rather well known. Around the core a special hardware must implement the functions mentioned above.

2.6. On the algorithms and programs which implement them.

Contrary to the usual situation during the development phase, the algorithms are maintained unchanged for long periods - let be only days - when running in real experiments; they run repeatedly every 10 μs. They must be implemented to achieve the fastest run and it is worthwhile to write them in assembler language if the gain in speed is significant.

3. The processor.

The flexibility we demand from the processor imposes a general purpose microprocessor but the fact we have a specific operation - the dot product - means we need a special facility for fast implementation of it. In general this dot product is just a step in computing a convolution so, in fact, a very fast convolver must be attached to the general purpose microprocessor. From the architectural point of view this means more than a coprocessor because the memory has to be accessed in a highly parallel way and a battery of multipliers is needed. Fig.4 shows a rather general structure which allows in the limit one dot product per machine cycle. The products - the intermediate results - can also be stored back in the memory if the needed outcome is a vector (Fig.5). The proposed architecture also provides fast pointwise transformations. The Annex explains the processor structure (Fig. 4 and 5).

Let us analyze a rather complex benchmarking algorithm from the point of view of computing power it needs : the algorithm for electron-pion separation in Spacal (the algorithm and the program in [3], Annex). There are 8 sums (trivial dot products) and 2 non-trivial dot products over the entire field (256 pixels) plus 5 scalar divisions, 4 scalar multiplications, and a set of 3 operations over the entire field (vector operations) : a comparison, a multiplication, and a nonlinear function (logarithm). The fastest way for the last operation is a LUT technique; to read and store the log we need once more the time to input the data. Suppose the degree of parallelism for the comparison and multiplication is the same as for the dot product. So the number of equivalent dot products rises to 13. Let consider the hypothesis that a battery of only 16 multipliers constitute the convolver (as we mentioned , there are TV real-time convolvers much bigger than 4x4). Then every operation on the 256 pixels field will be done in 16 steps. The total number of equivalent dot products amounts to 198. The machine must compute a dot product of 16 terms every machine cycle. An overhead of a few cycles must be added for every group of 16 terms products, let it be 50%. Then the total number of machine cycles amounts to 300. The scalar division needs about 30 cycles, i.e. 150 cycles are necessary for 5 divisions. We can suppose these operations (accomplished by the microprocessor) are done in parallel with the dot products. The same for LUT operations. Instead, we shall supplement a communication time estimated of the same order as the total computing time. Approximately 600 cycles result for the whole algorithm. Hence, the order of magnitude for the run time of this algorithm, with the slowest version of modern RISC microprocessors, 20MHz, is 600*50 = 30000 = 30 μs. Suppose the 10 μs between two events at the input of the second level triggering machine are fully used for input data. This means the total time - input plus processing - necessary for one region of interest of an event is about 40 μs in this example. Now suppose all the data pertaining to one event, 5 region of interest at most (as estimated in [2]) are fed in the same processor. It results 5*30 + input time = 160 μs. It means that at least 16 processors are needed in the farm.

We merely have done a coarse estimation of the needed number of processors. An extended analysis to other algorithms will be done to prove that, indeed, for the second level

triggering demands, a farm of 20-30 such processors will be always sufficient. Nevertheless the scalabilily of the machine insures the possibility to add more processors if necessary. We believe that with nowaday faster chips the number of processors could be even less than 10.

4. Conclusions.

One may sum up the reasons for the farm solution : The general purpose supercomputers cannot be used as a second level triggering machine because they do not have enough speed and the prices are extremely high. The massively parallel machines developed in the last decade show a firm evolution from the one bit processor to powerful many bits processor per node. *The fit of the architecture to the computing needs is essential for an efficient solution;* in our case the unit for processing is the processing of one region of interest, and actually there is little correlation between two successive units. *Optimizing the processing at the level of a region of interest* would need to exploit any kind of parallelism. The solution to surround a powerful microprocessor with special coprocessors and dedicated hardware *fitted for the identified computing needs*, appears far more efficient than simply multiplying the microprocessors constituting the "processor" (on the farm). Also developing the software for super-processors containing microprocessors is much more difficult than using existing software for the machine with two coprocessors.

Annex.

The architecture shown in the system block diagram of the processor suppose a multiple parallelism at the level of the processor blocks, namely :
a) A DMA-type LUT operation : memory LUT memory;
b) A multiple product/dot product operation;
c) A floating point operation;
d) Any other internal IU operation.

The parallelism can be achieved with modern circuits. Some key features in this respect are: a) the possibility of concurrent operation of master unit, coprocessor and floating point unit; b) the existence of dual port memory chips; c) the existence of fast 16*16 programable multipliers and fast large memories (64K). *As an example (only)*, the multipliers, fast access large memories, dual port memories, and SPARC RISC family from Cypress Semiconductor were taken.

The LUT is simply a 64K memory (e.g. two chips of 64K*8) and the MACs are also standard high speed chips. As for microprocessor we shall quote from DataBook [4]: "The CY7C601 integer unit supports a tightly coupled floating point interface and coprocessor interface that allows concurrent execution of floating point, coprocessor and integer instructions"; "The CY7C601 is the primary processing engine in the SPARC architecture, executing all instructions except for specific floating point and coprocessor operations. The CY7C602 FPU does its floating point calculations concurrently with CY7C601 IU. The architecture also allows for concurrent operation through the use of an optional second processor" (from User's Guide [5]). Regarding the CY7C130/131 dual ports memory: "Two ports are provided permitting independent access to any location in the memory".

The configuration aims at balancing the needs in all types of operations (LUT, vector/dot product, floating point) at the level of a "region of interest" and pushing the parallelism of the system (processor blocks) to its limits bearing in mind the chips' characteristics.

References

1. R.K. Bock et all. *Draft 3 RD-11 Status Report : Embedded Architectures for Second-level Triggering (EAST)* EAST note 92-05, 21 Feb. 1992.
2. R.K.Bock et all. *R&D Proposal Embedded Architectures for Second-level Triggering in LCH Experiments (EAST)* CERN/DRDC/90-56, 30 Oct. 1990
3. J.Badier, R.K.Bock, C.Charlot, I.C.Legrand *Benchmarking Architectures with Spacal Data* EAST note 91-10, 25 Nov. 1991.
4. DATACUBE *MaxVideo User's Manual.*
5. CYPRESS SEMICONDUCTOR *BiMOS/CMOS Databook.*
6. CYPRESS SEMICONDUCTOR *SPARC RISC User's guide.*

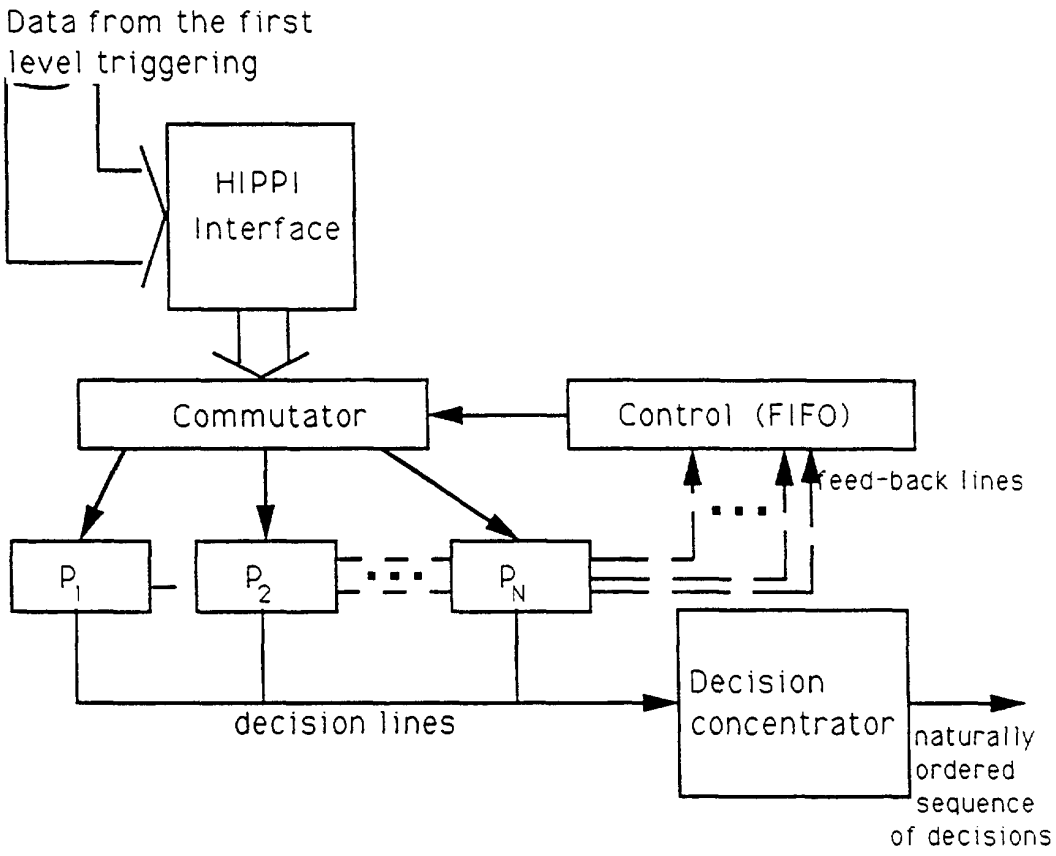

Data from the first
level triggering
HIPPI
Interface
Commutator
Control (FIFO)
feed-back lines
P1
P2
PN
decision lines
Decision
concentrator
naturally
ordered
sequence
of decisions

Fig. 1

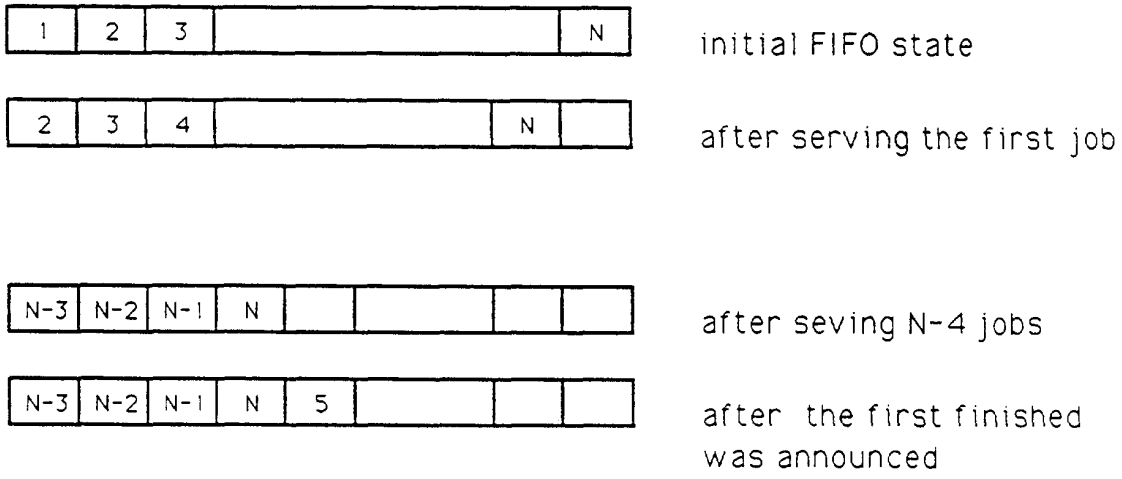
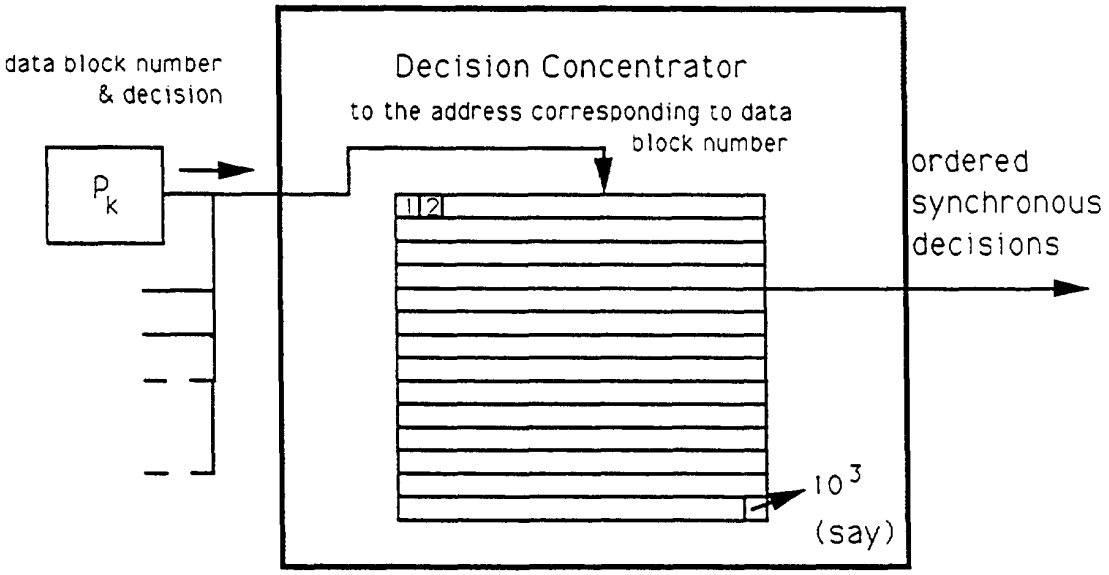


Fig. 2



data block number becomes the memory address for storing the decision message

Fig. 3

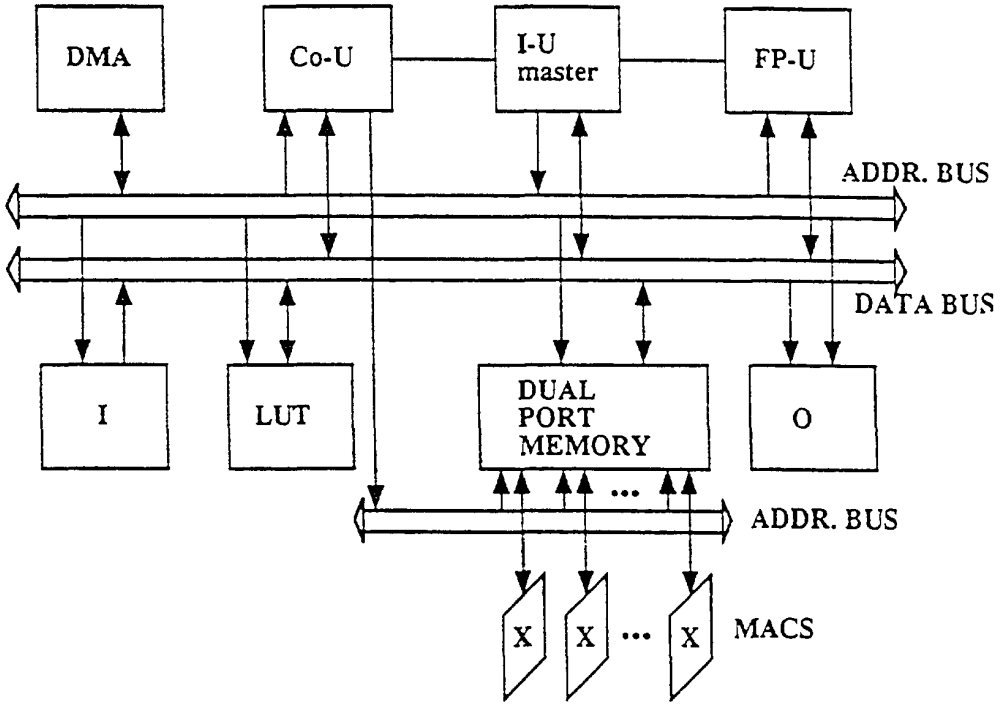


Fig. 4 System block diagram.
Floating point operation (through FPU)
Dot product/parallel products (through Co-U-MACS)
and LUT operations (through IU or DMA)
are done in parallel

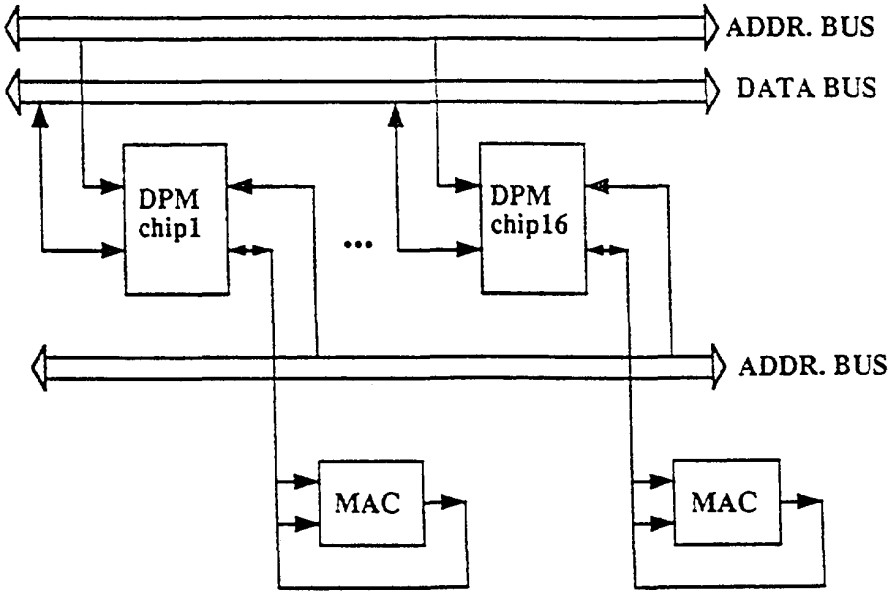


Fig. 5 Connecting MACs in the system