NIKHEF
NATIONAAL INSTITUUT VOOR KERNFYSICA EN HOGE-ENERGIEFYSICA

April, 1983 NIKHEF-H/83-6

FAMP: AN EXAMPLE OF MULTIPROCESSING IN HIGH ENERGY PHYSICS

L.O. Hertzberger, J.C. Vermeulen, L.W. Wiggers.
NIKHEF-H, P.O. Box 41882, NL-1009 DB AMSTERDAM, The Netherlands

Abstract:

Possible applications of multi microprocessor systems in the different areas of experimental High Energy Physics are considered. It is shown that these systems, of which the Fast Amsterdam Multi Processor (FAMP) system is an example, are potentially very powerful in on-line and in off-line applications. It is discussed how the potential advantages have been exploited in the FAMP system by a careful choice of hardware and software. An overview is given of the use of the FAMP system for second-stage triggering, on-line event filtering and monitoring purposes in the NA11, UA1 and JADE experiments and of other possible applications.

Contribution to the "Three day in-depth review on the impact of specialized processors in elementary particle physics", Padova, March 23-25, 1983.

1.0 INTRODUCTION

Present day high energy physics counter experiments aim for the detection of a large number of particles and for a complete and accurate measurement of their properties. Together with the availability of high beam intensities this causes large amounts of data to be handled at high rates which poses problems at all levels of data processing.

For reducing the dead time of an experiment and recording only the events of interest, selective triggering is required. Several trigger levels can be distinguished. The time available to take a decision and the complexity of the algorithm increases going from lower to higher levels. The lowest (zero) level decisions are in general performed by fast ECL type hardwired electronics. First level decisions may use complicated trigger matrices and special purpose micro-coded processors. A characteristic feature of this level is, that trigger parameters can be changed. Higher level trigger techniques will use processors, which are completely programmable and can execute rather complicated algorithms. At this level it is possible to apply various trigger algorithms without the necessity for hardware modifications.
Typically a zero level trigger takes a decision within 10, ns, a first level trigger has a decision time of 0.1 to 10 µs whereas higher levels need about 0.1 to 10 ms. Typical values of the corresponding reduction factors in the number of events to be processed are 10^3 - 10^5 for the lowest and the first level together and a factor varying between 2 and 100 for higher levels.

Only a small fraction of the electronic channels in an experiment contains valid data. Special-purpose processors can be used for sparse data scanning and data compression, before the data is actually transferred.

Modern experiments usually read the data from the different parts of the detector in parallel into a data buffer, before writing the data onto tape. This technique opens the possibility to apply on-line event filtering: one or more processors spying into the data stream can make the decision whether the event should be written to tape and flag the event in case of a positive (or negative) decision.

The large amount of electronic detector channels (typically 10^4 - 10^5) and the presence of support equipment such as power supplies, gassystems etc. creates a demand for sophisticated monitoring techniques. The monitoring task can be performed by a central computer, or by various (micro-) computers scattered over the experiment.

The off-line analysis of the data is costly in terms of computing power needed. The number crunching capabilities of an existing mainframe computer can in principle be extended by attaching to it one or more relatively cheap but fast processors.

It can be attractive to use multi-microprocessor systems for these different areas of dataprocessing because of the following features:

1 - The computing task can be split in a number of subtasks which can run concurrently.

2 - Present day microprocessors are powerful and relatively cheap.

3 - The processing power of a multi-microprocessor system is easily extendable.

4 - The possibility of constructing a redundant (and thus fault-tolerant) system.

5 - The simplicity of optimizing such a system, if constructed modulary, towards its application.

In this contribution an existing multi microprocessor system, the Fast Amsterdam Multi Processor (FAMP) system [1] is described. It is shown that systems as FAMP are potentially very powerful in the applications described before. After a description of the hardware and associated software of FAMP, an overview is given of the use of the FAMP system in the NA11 and UA1 experiments at CERN and in the JADE experiment at DESY. Finally, other possible applications of FAMP are indicated.

2.0 PARALLEL PROCESSING

For second level triggering, on-line event filtering and off-line number crunching the speed with which the algorithms are executed is of crucial importance. The execution speed of conventional microprocessors is not high enough to make their use feasible in most of the applications mentioned in the introduction. By constructing the hardware such that instructions and/or data can be handled in parallel, this limitation can be overcome:

- In large scientific computers the execution speed of a basically sequential machine is improved by the use of a number of directly connected but independent processing elements that all execute the same instruction stream and by the extensive use of instruction pipelining [2]. The speed gain is dependent on the type of algorithm to be executed. Algorithms frequently encountered in high energy physics, such as track reconstruction, that contain many conditional branches, are in general not able to make full use of the computing power of this type of architecture.

- So-called "Multiple Instruction, Multiple Data stream" (MIMD) architectures [3] make use of independent processors, each with a control unit that fetches the instructions from a memory attached to a particular processor or common to some or all processors in the system. Computing tasks in high energy physics can often be split up in a number of subtasks that can run concurrently (e.g. calculations of the energy deposit in a calorimeter and track finding in a drift chamber system). These types of problems can thus efficiently be handled by a MIMD type of machine.

A distinction can be made between MIMD machines in a number of ways, e.g. according to the number of busses or to the degree of specialization of the various processors, etc.

Here we distinguish two classes of systems:

- loosely coupled systems (networks),
- tightly coupled systems (multiprocessor systems).

In loosely coupled systems the processing systems are interconnected via serial lines, varying in speed from slow RS232 lines to high speed local area networks like Ethernet[‡][4]. A characteristic feature of a loosely coupled system is that the processing systems are physically distributed; they can be located in different rooms or buildings. The processing systems perform in general completely different tasks and interchange only a limited amount of information. A loosely coupled system can be used for monitoring the various parts of an experiment.

Tightly coupled systems are characterized by high-speed parallel connections between processing elements such as a time-shared common bus. The processing elements cannot be placed far apart because of the limitation set by their type of interconnection:
they will be located in the same or nearby crates. This kind of system is well suited for tasks in which all processors have to work closely together. Tightly coupled and loosely coupled structures are the two extremes of multiple processor organization. Structures that combine the better qualities of each, are more suitable for microprocessor applications. Second-stage triggering, on-line filtering and off-line number crunching can in principle be handled efficiently by such a moderately-coupled, so-called Multiple Task Multiple Data stream system (MTMD)[3]. By using relatively cheap but powerful and widely used microprocessors in the processing elements and by adopting a modular construction such a system can also be cost-effective and flexible in its use. The FAMP system is an example of such an MTMD system, designed along these guidelines.

3.0 FAMP HARDWARE

3.1 System overview

The hardware of the FAMP system is constructed in such a way that the system can be easily adapted and optimized towards the particular requirements of the experimental situation:

(i) The system consists of modules that fit mechanically into a CAMAC crate. From the CAMAC standard only the CAMAC power supply lines are used. A special backplane, forming the internal system bus, has been designed for communication between modules. A processing cell consists of a CPU module together with other system units like memory and interface units. These units communicate with each other via the internal system bus. Only one CPU module can be connected to the internal bus; more than one CPU module can fit into one crate, provided that the internal bus is subdivided and properly terminated.

‡ Ethernet is a trade mark of Xerox Corporation.

(ii) Processing cells are connected with each other via dual port memories, as illustrated in fig. 1. This figure shows that the hardware of the system allows for multiple trees and interconnections with a processing cell (the "supervisor") at the root of each tree. The dual port memories are constructed as separate modules and have access to the internal system bus via the backside connector and to the so-called external bus via a frontside connector. The external bus consists of a 64 lines flat cable. The CPU module has via a front panel connector access to the external bus. The coupling between processing cells can thus be realized using the external bus as illustrated in fig. 2.

3.2 The CPU module

The CPU module (fig. 3) consists of two boards that communicate via the internal system bus. One board contains the CPU and necessary support chips whereas the other mainly contains memory.

For the CPU the Motorola MC68000 [5] processor is used in view of its advanced architecture and its speed. The 68000 has a 16 bit wide data bus and a 24 bit wide address bus. It contains 8 address and 8 data registers, each 32 bit wide, while the instruction set supports operations on bytes (8 bits), words (16 bits) and long words (32 bits). The cycle time of 125 ns provides a minimum instruction time of 500 ns.

The local memory in the CPU module has a capacity of up to 32 Kbyte. It is built out of 16 Kbyte static (150 ns access time) RAM and 16 Kbyte EPROM. Part of the EPROM is used to store the operating system software. Jumpers on this board offer the possibility to exchange EPROM for RAM.

Two RS232 lines are available. One can be used to connect a terminal, the other to down-line load user programs.

3.3 The dual port memory module

The dual port memory module is designed to connect two processing cells, as is discussed in 3.1. Two independent ports to the memory are available, connected to the internal and external bus. To prevent clashes, arbitration logic handles multiple access requests from both ports. The memory has a capacity of 16 Kbyte static (150 ns access time) RAM, which can be placed at different positions in the memory maps of the two connected processing cells.

3.4 Memory modules

As an extension of the CPU-module memory a 256 KByte dynamic RAM and a 64 KByte static RAM module [6] have been developed. The memory boards designed for the CPU unit can also be used as separate modules.

3.5 Interfaces

Various interface modules have been developed:

- NIM I/O module
  This module provides the facilities to exchange input and output signals with the other decision logic in the experiment. It has 24 input lines and 4 output lines, which can be read or set individually. The NIM signal convention is followed. Two of the input lines can act as vectored interrupt lines. They can be masked via an internal register.

- ROMULUS output buffer and REMUS compatible interface
  For the data transfer to the host computer the ROMULUS [7] read-out convention is used. An output buffer has been designed that behaves like a Read Only Crate Controller (ROCC). For the host computer the whole FAMP system is seen as a slave in a ROMULUS branch. The CPU module writes data into the memory of the interface (8 KByte static RAM), which is subsequently read by the ROMULUS branch. Data protection is guaranteed by a semaphore register, accessible by FAMP as well as by the ROMULUS branch. The output buffer is at present being re-designed towards the REMUS [8] specifications which permits the writing of control functions. 4.1

- CAMAC interface
  A prototype of an A2-type Crate controller has been constructed. Via the external bus it is connected to a CPU module or to an interface in the FAMP system. The combination can work as a stand-alone system. It is foreseen to build an auxiliary version of the controller as well.

- Experiment dedicated interfaces
  To interface the RMH-read-out system in the NA11 and the MTD system in the UA1 experiment, special controllers have been constructed (see section 5). They write the reordered data in intelligent memory modules, which support some special hardware functions.

3.6 Bus display

The bus display module indicates the states of the lines of the internal bus and therefore can be used as a debugging tool. It can trap on a selected bus address and has the possibility of single stepping through bus cycles.

4.0 FAMP SYSTEM SOFTWARE

The system software supports separate processors as well as the multiprocessor environment. For the time being the software is written in assembler. However, parts may later be rewritten in a higher level language. The system may be used in the "data acquisition" mode as well as in the "operating system" mode. The data acquisition mode supports the running of time critical user programs, e.g. trigger programs, whereas the operating system mode supports the not time critical functions, e.g. running of testprograms and system monitoring tasks. As a consequence user programs must perform with absolute priority, whereas the operating system tasks have to run as background programs. The modes are selected by external interrupts.

Because the systèm has to run in two distinct modes, special software solutions are required, for instance concerning the management of the data stream through the dual port memories and the interaction of the user programs with the system. In the FAMP operating system the interrupt facilities are provided by the system software, running in the operating system mode.

4.1 The operating system mode

The operating system is in principle the same for all processing cells. However, one of the processors can act as "supervisor" while the others act as "slaves". The supervisor interprets the commands given at a connected terminal and transmits them, if needed, to the slaves.

The operating system consists of two parts:

(i) The kernel, which is equal for all processors, provides the elementary system functions, such as system initialization and exception and interrupt processing. System initialization includes the setting of some individual vectors and I/O initialization.
The software is written such that in the operating system mode the MC68000 is always in the "supervisor state", whereas during the data acquisition mode the processor is in the "user state". In the supervisor state priviliged instructions are allowed, in the user state they are not.
A number of operating system routines can also be called by user programs. As user and supervisor have separate stackpointers, the protection of the stacks used by the operating system programs and those handled by the user program is guaranteed. The switch from user to supervisor state is made automatically in any of the following circumstances:

- a user program calls on the operating system to execute some function, requiring the use of a privileged instruction. Such a call is termed a supervisor call (the TRAP instruction is used for this purpose),
- an interrupt occurs,
- an error condition occurs in a user program.

The switch from supervisor state to user state goes through a privileged instruction.
The kernel monitors also the behaviour of the various processing elements. For example, as a protection against hang-up, a time-out interrupt is generated, when one of the processing cells is not responding within a certain time limit.

(ii) The remaining part of the operating system contains the more elaborate functions, like: 5.0

- hardware testprograms,
- debug facilities,
- down-line loading facilities,
- the command interpreter,
- communication facilities between processing elements.

The hardware test programs check the behaviour of memories and interfaces.

Debug facilities are for instance: displaying and/or changing of memory and register contents (including the stack pointer), and instruction by instruction tracing.

Down-line loading of user programs can be done via a serial link from the host computer to the supervisor and/or slaves. The programs can also be transmitted from the supervisor to the slaves via the dual port memories.

The command interpreter of the supervisor allows full control of the microprocessor-system, while that of the slaves only gives local control. 5.1

The communication between supervisor and slaves is accomplished by exchanging messages and associated data. Messages are written in a dual port memory module by one processor and read by another. The main problem is the synchronisation of the tasks running in the various processors. Moreover, the software has to control two distinct data flows, which originate from two asynchronous processes, i.e. the data of the experiment managed by the user programs and the information for the interprocessor communication managed by the operating system. Since these processes are not synchronized and both streams have to pass the same dual port memory, a problem arises when two processors want to write or read at the same memory location concurrently (e.g. the mutual exclusion problem). To solve this problem, the fraction of the dual port memory used by the operating systems is divided in blocks. The data structure used to manage these blocks consists of three circular linked lists (figure 4). One list contains messages. The other two lists keep track of free memory blocks, one is used for free message space, the other for free data space. When data is associated with a message, a pointer in the message block will point to the data block, which is released from the circular linked list of free data blocks. The blocks are accessed via the pointers. Insertion and deletion of messages and linking data to a message is a matter of changing pointers. The access to the pointer is protected by a semaphore. The testing, and when not set, the setting of a semaphore, must be an indivisible operation. The MC68000 provides such an instruction, which makes the processor suitable to operate in a multiprocessor environment.
An advantage of this approach is that it allows dynamical dual port memory management without the need of garbage collection. Additionally the number and sizes of the two kind of blocks can be specified during system generation time.

5.0 APPLICATIONS OF THE FAMP SYSTEM

The FAMP system has been designed for second stage triggering with in mind that for applications like on-line and off-line number crunching limited changes in the system would be needed. The configurations in the NA11 [8] and UA1 [9] experiment are used as second stage trigger systems, whereas the system in JADE [10] is ment to be an on-line filter. Additional requirements for a multi microprocessor system in order to be used as off-line number crunching facility will be discussed at the end of this section.

It has to be stressed, that these applications are in different stages of realisation. In the NA11 experiment FAMP has been used for data taking: the data are presently being analyzed. In the UA1 experiment the final configuration of FAMP has not yet been implemented. With the present set-up trigger programs are tested. The hardware for the JADE experiment is ready, the software has been partly tested.

5.1 FAMP in NA11

FAMP was used in a study of inclusive ϕ-meson production [8] in the NA11 experiment, to select $\phi \rightarrow K^+K^-$ decays by determining the charged K-meson trajectories and calculating the K^+K^- invariant mass. FAMP is used in conjunction with a set of proportional chambers and scintillator and Cerenkov counter hodoscopes in the spectrometer of the ACCMOR collaboration at the CERN-SPS. From the coordinate information of the proportional chambers the FAMP system calculates:

(a) The momenta of the K-meson candidates, checking whether they fall within the momentum band set by the Cerenkov counter thresholds.

(b) The invariant mass of the K^+K^- pairs.

Events fulfilling the criteria of this second stage trigger are read out by the NORD-100/500 data taking computer and recorded onto tape. For the read-out of the proportional chambers RMH-compatible D16/D32 Plessey modules are used. At the end of the RMH-chain a special module (Bypass Unit) gives control over the data stream either to the FAMP system or to the NORD computer. When the FAMP system has control, a purpose-built controller and concentrator (COCO) [9] reads data, reformats it and transfers the reformatted data to intelligent data-memory modules (PRIME) [9] connected to each processor. For the data bus the daisy chain concept is applied. In order to reduce the computing load of the microprocessors, special functions are implemented in the PRIME. They allow a sequential read-out of the data in the planes and the searching of a hit on a given wire number with a certain tolerance.

The FAMP configuration consist of one supervisor and two slaves. Each processing element contains a CPU module, two 32 KByte static RAM modules and a PRIME module (fig. 5).

The ROMULUS output-buffer in the supervisor crate is used to transfer data to the NORD on-line computer.

Processing of an event is started when the busy signal of the COCO disappears. This indicates that the read-out has been finished. When either one of the three processors decides that the trigger criteria cannot be fulfilled, an interrupt is provoked. Subsequently all processors are stopped and the read-out electronics is cleared by a NIM fast-clear signal. When the event is accepted, the NORD computer is triggered to read out all experimental data, bypassing the FAMP system. Intermediate results of the three processors are stored in the ROMULUS output buffer. They are used for debugging and monitoring of the FAMP system by displaying all intermediate results, like the K^+K^- mass 5.2
distribution, by the NORD on-line computer.

The momenta of the particles are determined using two views, 4 planes in total, of the proportional chambers. These data are processed by the two slaves in parallel. It is checked rougnly whether the proportional chamber hits are correlated with the hodoscope pattern which produces the fast trigger. If either of the slaves finds an insufficient number of track candidates a veto signal is produced. The coordinate and momentum information as calculated by the slaves is stored in the dual port memories. Subsequently the supervisor-processor scans the two sets of momenta corresponding with the two views for possible equal values. If such a momentum pair is found the processor searches with a hardware function in the PRIME for a possible correlated hit in a fifth plane. Also a more accurate check is made of the correlation with the Cerenkov information. When all track candidates are dealt with and a sufficient number of negative and positive particles have survived, a loop is performed over all combinations of oppositely charged particles to determine their invariant mass.

In all processing steps extensive use is made of large look-up tables; 32 KByte for each of the slave processors and 60 KByte for the master-processor.

The system has run succesfully from June till September 1982 for a total of 25 days. Various trigger conditions (K^+K^-, ϕ, $2K^+K^-$, $\phi\phi$) have been used. In total, 7 million events have been written onto tape. The largest statistics was collected with a single-phi trigger using a 100 GeV/c negative beam incident on a Be target. In the following, some results obtained in this run are presented.

A large percentage (42%) of the triggers is rejected in an early stage by the slave processors. Only 5% of the triggers is eventually accepted.
The mean reconstruction time for an accepted event is 750 μsec, for all events it is 450 μsec. The time distributions for the various event categories are shown in fig. 6.
In fig. 7 the K^+K^- invariant mass spectrum is plotted as calculated by the FAMP and displayed on-line. 5.2.

The second-stage trigger accepts about 20% more events than the off-line analysis program does, if similar selection criteria are applied. This is due to slightly relaxed trigger requirements and to ghost tracks found by the FAMP reconstruction program. On the other hand the system does not find all good events. The hit efficiency in a proportional chamber plane is 98%. Demanding 5 planes be hit for a track, causes a loss of 18% of

good K^+K^- events. The track finding efficiency of the FAMP when a hit in every plane has been found, amounts to 97%.
The dead-time without the FAMP system is fully determined by the CAMAC read-out time of the electronics. For a mean number of 4400 bytes/event of data 10 ms is needed. With a trigger rate of 3700/sec this gives 2.4% sensitive time of the spectrometer. Using the second-level trigger system, this improves to 25%. However, since it introduces an inefficiency of 23%, the effective gain in sensitivity is 0.77 x 0.25/0.024 = 8.0. This results in the recording of ten genuine phi-events per SPS-burst of 2.6 seconds.

5.2 FAMP in UA1

The UA1 [10] experiment studies with a multi-purpose detector $p\bar{p}$ interactions at the CERN SPS collider. The FAMP system will be used for triggering on high energy muons which may signal the production of W^+, W^- or Z^0 bosons. The muons are detected by a system of chambers consisting of drift tubes, about 6000 in total, which cover the full 4π solid angle [11]. The drift tubes are arranged in sets of four planes. Two perpendicular views are available in each set. The iron shielding between these chambers and the interaction point suppresses the enormous hadron background. A further reduction of the background is obtained by applying a direction criterion: a fast muon points to the vertex while hadrons and low energy muons will scatter over large angles in the magnetic field and in the material between the interaction point and the muon chambers.

The direction criterion is firstly used in the first level fast trigger. From the positions of the drift tubes that were hit by the particle a direction accuracy of about 300 mrad is obtained. The trigger decision takes only 1 µs, while the time in between beam-crossings is of the order of 4 µs. Consequently, no extra dead-time is introduced.

The second level muon trigger, as to be realized with the FAMP system, uses the drift times, which improves the directional accuracy to 30 mrad. The background is expected to be reduced by a factor of 5. An acceptable decision time for this trigger would be about 200 µs.

The second level muon trigger is one of several possible trigger schemes. The coordination of the various triggers is performed by the central trigger processor.

In the following the hardware configuration of the final second level trigger for high energy muons and of the set-up as used during the last period of data-taking (september-december 1982) are described.

5.2.1 The final FAMP set-up

The central trigger processor interrupts FAMP when it has selected an event. Via a bypass unit in the "Multiple Time Digitizer" (MTD) [12] system connected to the muon chambers the FAMP system can read-out the data. The read-out is done by 6 slave processors (fig. 8, 9) via reordering processors. Pairs of slaves work together on the two independent views, where each slave handles one projection. All slaves perform the same algorithm, written in assembler. This algorithm removes the left-right ambiguity of the position measurement by expressing the drift times in bit patterns and subsequently shifting the patterns of the four planes as a function of the positions of the drift tubes such that for a track pointing to the vertex the bit patterns of the planes overlap. Information from the 1st level trigger directs the search done by the 2nd level trigger.

The results from this procedure are placed in the dual port memories. The supervisor processor combines all results and communicates the final conclusion to the central trigger processor. If FAMP finds one or more high energy muon candidates or if the central trigger processor decides to take the event anyway, the bypass unit in the MTD system gives the NORD 100/500 data taking computer the opportunity to read out the muon chamber data. After this the NORD reads the track information obtained with the FAMP system from the ROMULUS output buffer.

In the supervisor a special program package, written in PASCAL, is provided to use FAMP for monitoring purposes. This program becomes active during the waiting time for the next event. It produces histograms of wire, time and track distributions. This information can be displayed on the terminal connected to the supervisor. In this way irregularities in the response of the chamber systems can rapidly be spotted. Before and after a period of data taking FAMP provides facilities for testing the MTD's and the chambers.

5.2.2 The set-up in 1982

The system consisted of 2 slaves and a supervisor. Because the MTD read-out bypass unit was not available the MTD's were read-out via the FAMP system. The test and histogram facilities were used but the FAMP trigger was invoked only in off-line mode. The decision time was 2 ms in stead of the 200 μs foreseen for the final set-up. This is because the reordering processors were not yet available, so that consequently the reordering had to be done by software.

5.3 FAMP in JADE

In the JADE [13] experiment FAMP is foreseen to be used as an on-line filter. The configuration will consist of a supervisor with a number of slaves. The data is first stored into the memory of the NORD 100/500 computer and subsequently read by the supervisor using a special designed interface via which the supervisor can address part of the NORD memory. The supervisor sends the data to the dual port memory of a free slave. After the data is transported, a flag in the dual port memory is set indicating that data processing by the slave can start. All slaves are executing the same program, looking for one good track coming from the vertex. If such a track is found the supervisor is informed. The latter than sends the event back to the NORD computer where it is written on tape. The track reconstruction program is written in FORTRAN. First tests show that an average processing time of 1 sec per event can be obtained.

5.4 FAMP as a general purpose facility

When systems like FAMP are used for general purpose applications like number-crunching and interactive graphics, it is desirable that user programs as running on large scientific computers (CDC, IBM) can be transferred with as few modifications as possible. Most of these programs are written in FORTRAN and extensively use floating-point operations. Test with analysis programs of JADE and NA11 show that the high level language compilers available (like FORTRAN and C) do not produce very efficient code.
Moreover the lack of floating-point instructions in the current versions of the MC68000 causes, that time-consuming software alternatives have to be applied. This drawback can be overcome when special hardware will be available. For instance, for the 32-bit version of the MC68000, the MC68020, a floating-point co-processor is foreseen.

It is attractive to have a run time system allowing for integration of the software development and the actual execution environment. Therefore an operating system, written in C, that uses one central filing system is developed. For a user the multi processor system is accessible via a terminal on a central processor, running the UNIX[‡] operating system. This processor, with the central filing system, has memory management, and a number of peripherals attached. The communication between tasks executing on the various processors as well as the manipulation of files is realized via message passing primitives [14]. Message passing excludes the need for common addressable memory in between processors. It makes the run time system suited for a network environment. For the interrupt handling on all processors except the UNIX processor a solution like the ADA "ACCEPT" statement [15] is adapted. First tests have been performed on a 68000 based UNIX system with a serial link to a three processor FAMP configuration.

6. Conclusions and further developments

In this contribution it has been discussed how multiprocessor systems and in particular FAMP can form an attractive solution to problems arising in various areas of High Energy Physics data processing. Since more powerful and nevertheless cheap microcomputer boards are appearing on the market and bus standards, like VME [16], are emerging, it will become more and more attractive to apply this kind of systems.

The drawback of inefficient compilers can be expected to be overcome in the near future because of the pressure of a world wide market. Also it is probable that compilers for languages like MODULA and ADA which have special facilities for efficient program development such as information hiding and generics, will be released.

‡ UNIX is a trade mark of Bell Laboratories.

Acknowledgements

The FAMP project has benefitted from the work of and discussions with a large number of people. The hardware was designed among others by G. Kieft, T. Pauw, M. Ho, H. Groenstege and G. Evers at NIKHEF and B. van Koningsveld and C. Engster at CERN. Important contributions to the development of the FAMP software were made by D. Gosman, M. Schoorel and F. Tuynman at NIKHEF. The application of the FAMP system in the UA1 experiment was possible thanks to the help of K. Eggert and co-workers (in particular K. Giboni) of the Aachen group in UA1, D.J. Holthuizen from NIKHEF and D. Samyn from CERN. The application in the NA11 experiment was realized by W. Hoogland and co-workers of NIKHEF. We are also grateful to E. Pietarinen and co-workers of the Helsinki/DESY group in the JADE collaboration for the information they gave us concerning the application of FAMP in JADE.

REFERENCES

1. L.O. Hertzberger, Comp. Phys. Comm. 26 (1982) 79.
2. A.S. Tanenbaum, "Structured Computer Organization", Prentice Hall, 1976, p.24.
3. E.T. Fathi, M. Krieger, Computer, 16, No.3, March, (1983) 23-32.
4. R.M. Metcalve, D.R. Boggs, Comm. ACM, 19, No.7, July, (1976) 395-404.
5. Motorola, MC68000 user's manual UM(AD2) 1980.
6. E. Pietarinen, private communication.
7. P.J. Ponting, CERN EP/80-01.
8. C. Daum et al., NIKHEF-H/83-2 (1983), submitted to the Wire Chamber Conference, Vienna, 1983. To be published in Nucl. Instr. and Meth.
9. H.L. Groenstege, G.J. Evers, NIKHEF-H/82-26 (1982).
10. A. Astbury et al., CERN SPSC/78-19 (1978).
11. K. Eggert et al., Nucl. Instr. and Meth. 176 (1980) 217.
12. K. Eggert et al., Nucl. Instr. and Meth. 176 (1980) 123.
13. W. Bartel et al., Phys. Lett. 88B (1977) 171; 92B (1980) 206; 99B (1981) 277.
14. W. Morven Gentleman, Software Practice and Experience, 11 (1981) 435.
15. J. Welsh and A. Lister, Software Practice and Experience, 11 (1981) 257.
16. Motorola, VME Bus Specification Manual M68KVMEB (D1), 1981.

FIGURE CAPTIONS

Fig. 1	Two examples of connections of processing cells.
Fig. 2	Connections at bus level.
Fig. 3	Block scheme of the CPU module.
Fig. 4	Operating system message structure.
Fig. 5	The FAMP configuration in the NA11 experiment.
Fig. 6	Distributions of the reconstruction times for different event categories in the NA11 experiment. Stage 1 stands for looking for K^+K^- candidates. Stage 2 represents the calculation of the invariant mass.
Fig. 7	K^+K^- mass distribution as calculated on-line by the FAMP system in NA11. The peak originates from ϕ-mesons (with a mass of 1.02 GeV/c^2) decaying into a K^+ and a K^- meson.
Fig. 8	Organization of the read-out of the muon chambers of the UA1 experiment.
Fig. 9	The future FAMP configuration for the UA1 experiment.

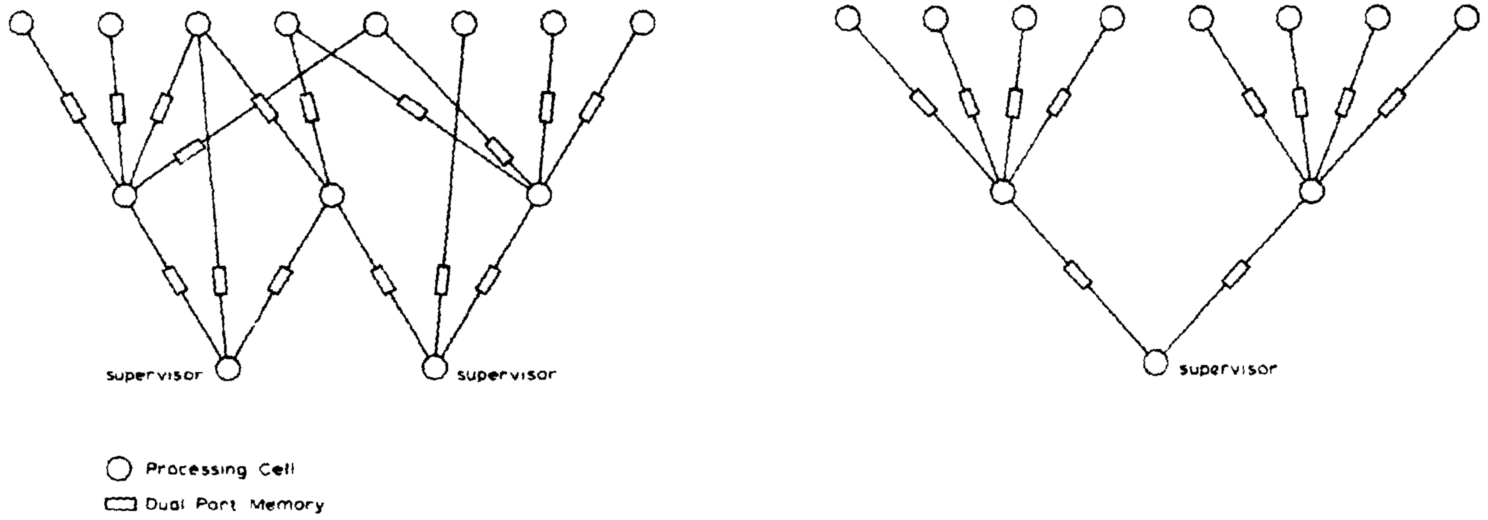

supervisor
supervisor
supervisor
Processing Cell
Dual Port Memory

Fig. 1

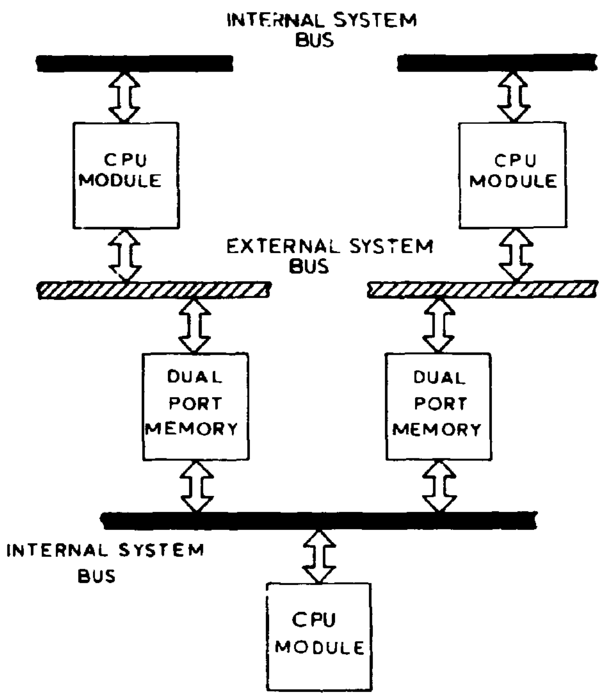

INTERNAL SYSTEM BUS
CPU MODULE
CPU MODULE
EXTERNAL SYSTEM BUS
DUAL PORT MEMORY
DUAL PORT MEMORY
INTERNAL SYSTEM BUS
CPU MODULE

Fig. 2

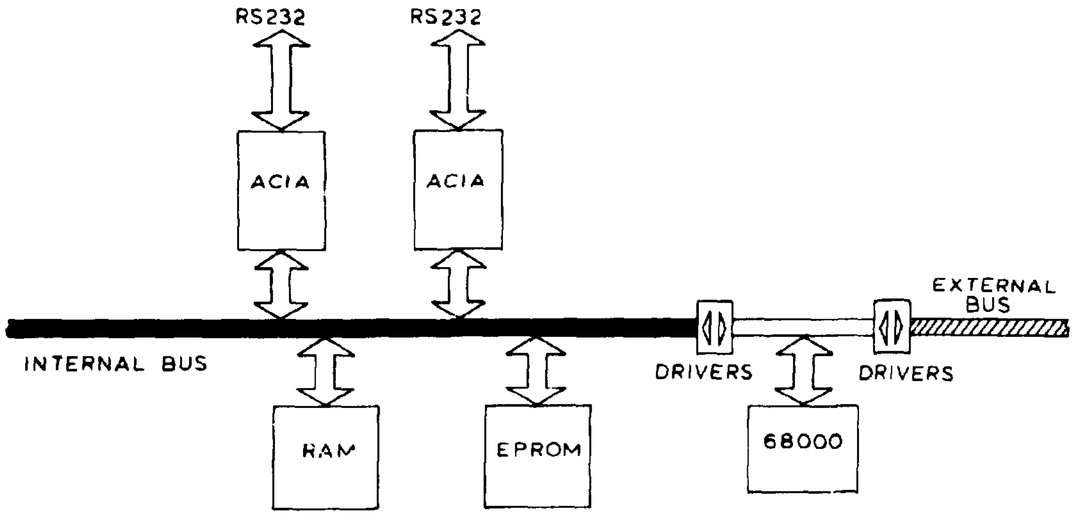

RS232
RS232
ACIA
ACIA
EXTERNAL BUS
INTERNAL BUS
DRIVERS
DRIVERS
RAM
EPROM
68000

Fig. 3

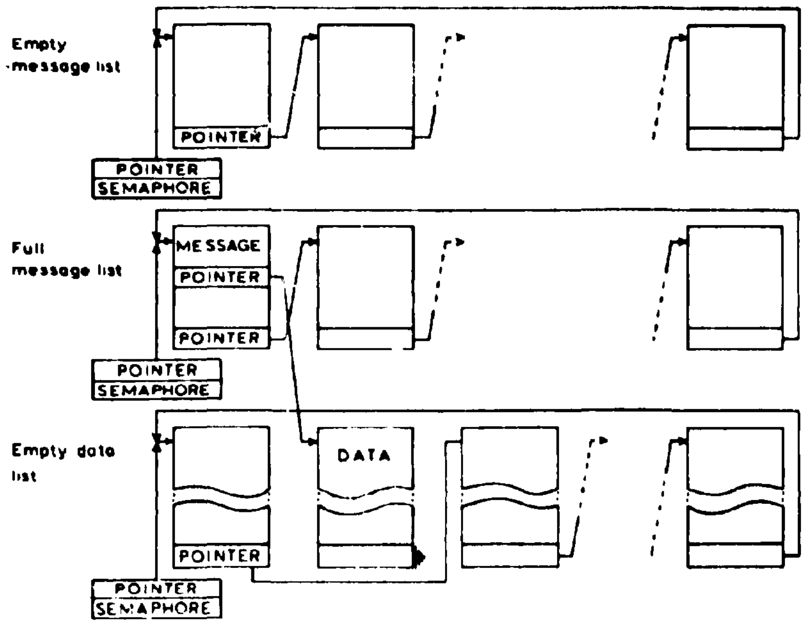
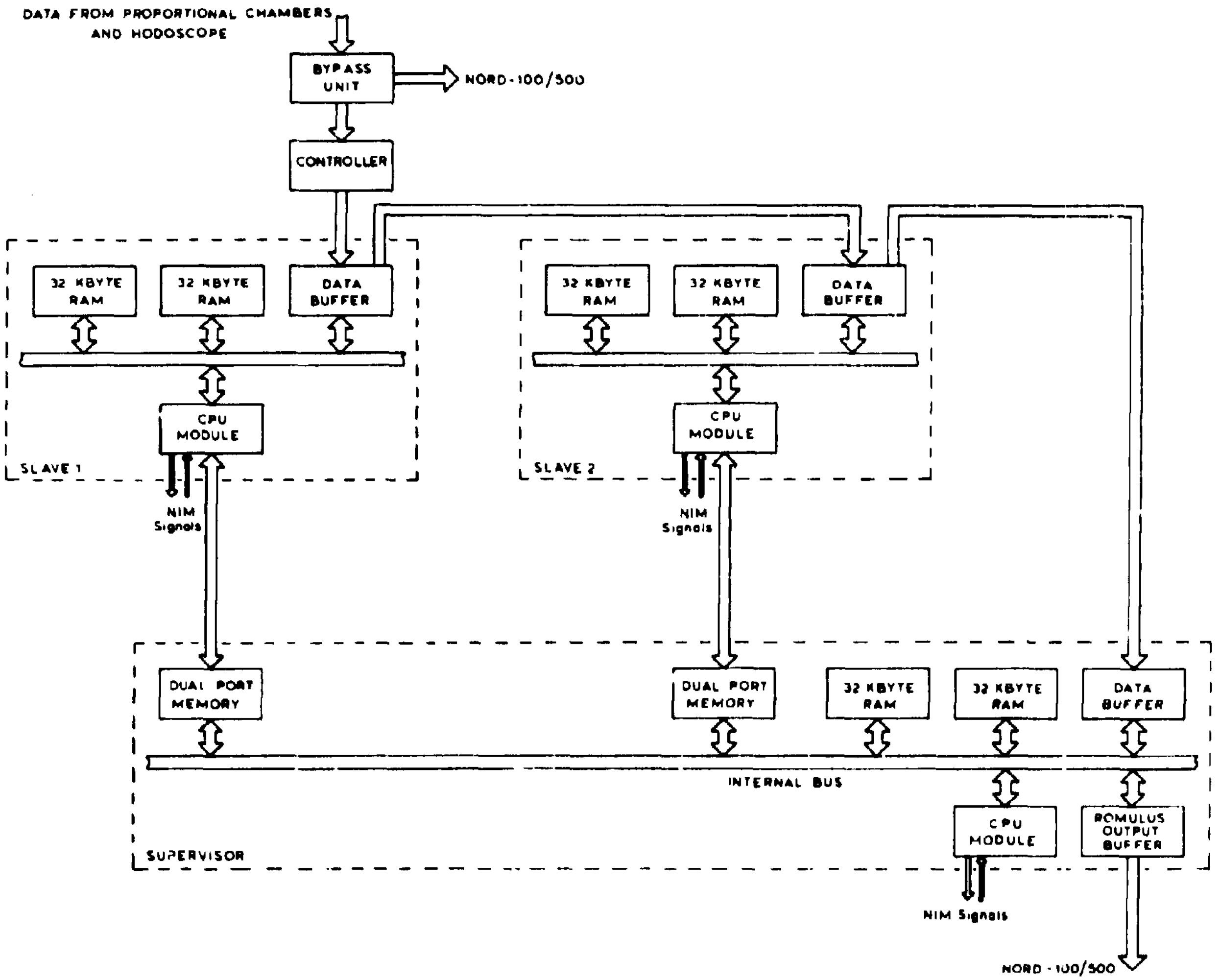
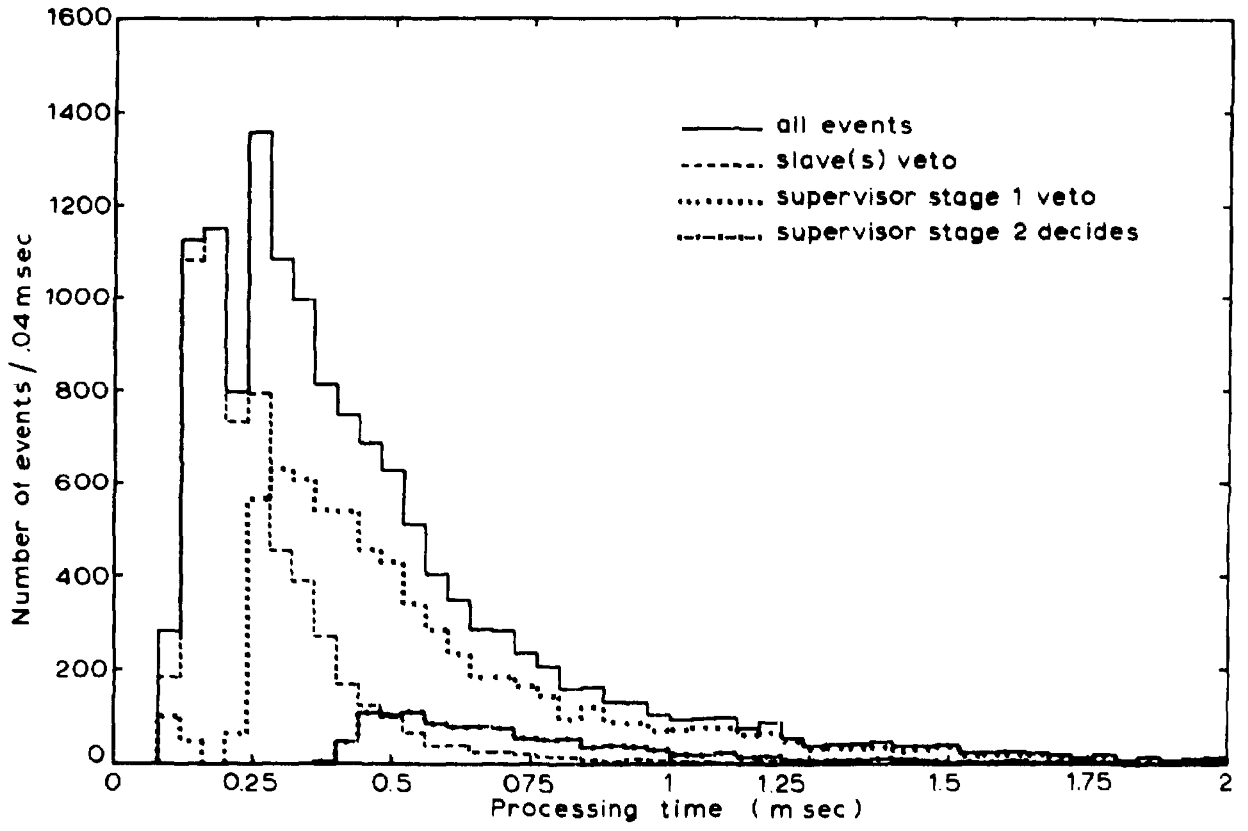

Empty message list
POINTER
POINTER
SEMAPHORE
Full message list
MESSAGE
POINTER
POINTER
POINTER
SEMAPHORE
Empty data list
DATA
POINTER
POINTER
SEMAPHORE

Fig. 4



DATA FROM PROPORTIONAL CHAMBERS AND HODOSCOPE
BYPASS UNIT
NORD-100/500
CONTROLLER
32 KBYTE RAM
32 KBYTE RAM
DATA BUFFER
CPU MODULE
SLAVE 1
NIM Signals
32 KBYTE RAM
32 KBYTE RAM
DATA BUFFER
CPU MODULE
SLAVE 2
NIM Signals
DUAL PORT MEMORY
DUAL PORT MEMORY
32 KBYTE RAM
32 KBYTE RAM
DATA BUFFER
INTERNAL BUS
CPU MODULE
ROMULUS OUTPUT BUFFER
SUPERVISOR
NIM Signals
NORD-100/500

Fig. 5



1600
1400
1200
1000
800
600
400
200
0
Number of events / .04 msec
0
0.25
0.5
0.75
1
1.25
1.5
1.75
2
Processing time (m sec)
all events
slave(s) veto
supervisor stage 1 veto
supervisor stage 2 decides

Fig. 6

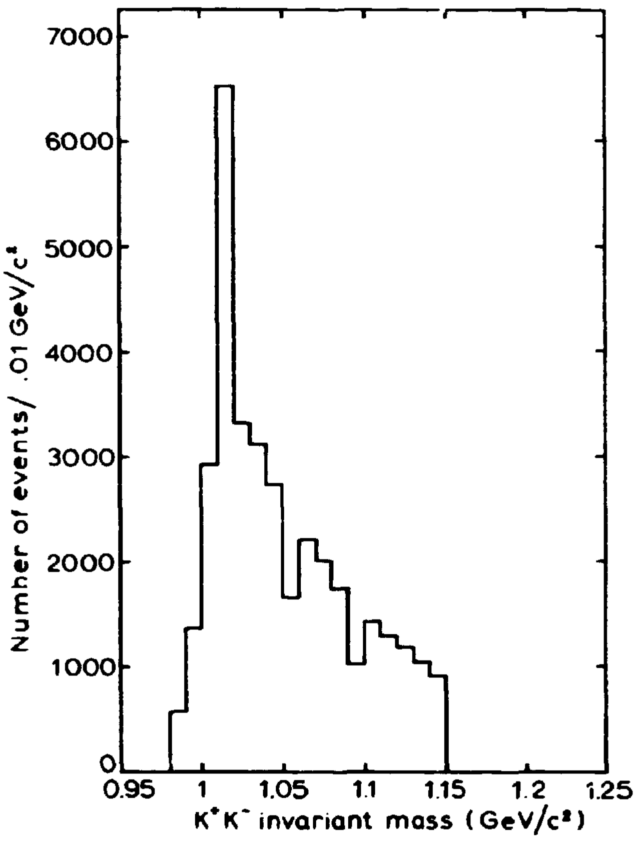

7000
6000
5000
4000
3000
2000
1000
0
0.95
1
1.05
1.1
1.15
1.2
1.25
Number of events/ .01 GeV/c²
K⁺K⁻ invariant mass (GeV/c²)

Fig. 7

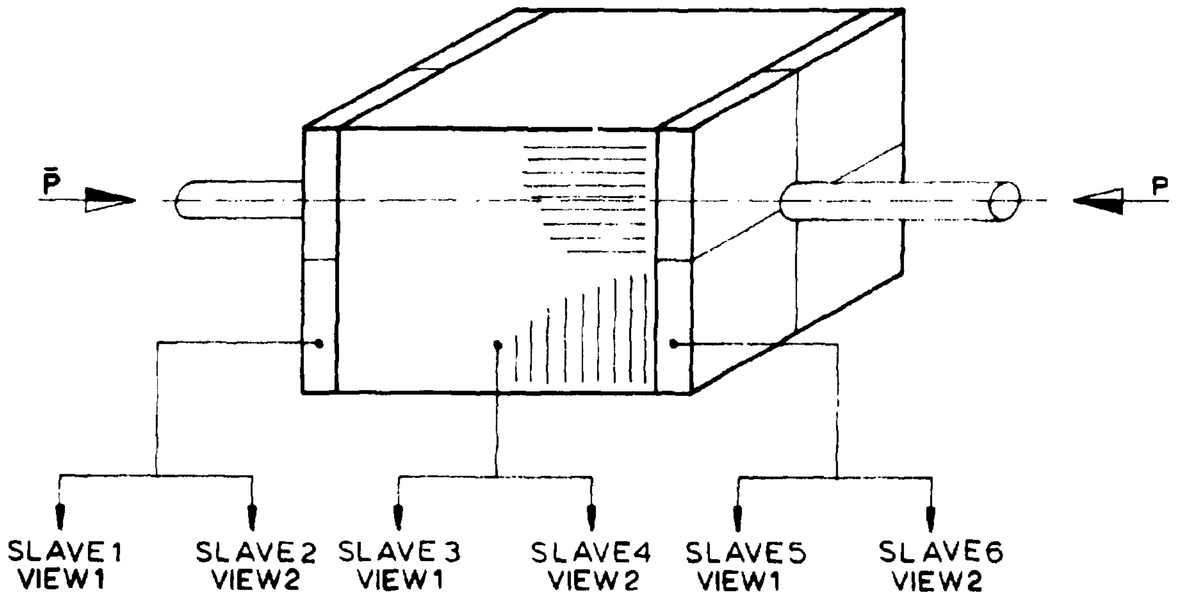
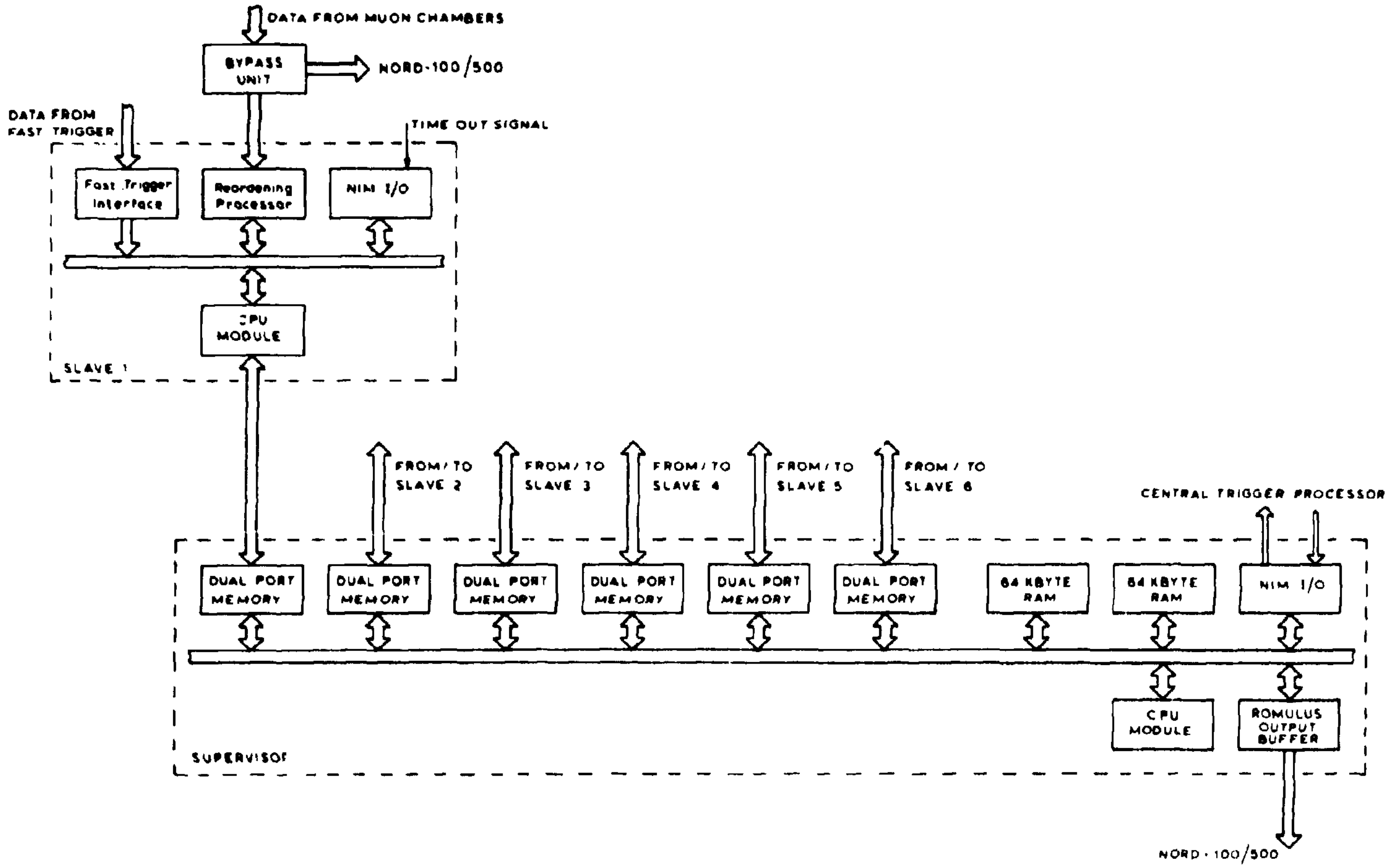

$\bar{P}$
P
SLAVE1 VIEW1
SLAVE2 VIEW2
SLAVE3 VIEW1
SLAVE4 VIEW2
SLAVE5 VIEW1
SLAVE6 VIEW2

Fig. 8



DATA FROM MUON CHAMBERS
BYPASS UNIT
NORD-100/500
DATA FROM FAST TRIGGER
TIME OUT SIGNAL
Fast Trigger Interface
Reordening Processor
NIM I/O
CPU MODULE
SLAVE 1
FROM/TO SLAVE 2
FROM/TO SLAVE 3
FROM/TO SLAVE 4
FROM/TO SLAVE 5
FROM/TO SLAVE 6
CENTRAL TRIGGER PROCESSOR
DUAL PORT MEMORY
DUAL PORT MEMORY
DUAL PORT MEMORY
DUAL PORT MEMORY
DUAL PORT MEMORY
DUAL PORT MEMORY
64 KBYTE RAM
64 KBYTE RAM
NIM I/O
CPU MODULE
ROMULUS OUTPUT BUFFER
SUPERVISOR
NORD-100/500

Fig. 9