A HIGH SAFETY MULTI-MICROPROCESSOR ARCHITECTURE

Y. DESWARTE

COMPAGNIE INTERNATIONALE POUR L'INFORMATIQUE

DIVISION MILITAIRE SPATIALE ET AERONAUTIQUE

VELIZY 10 Avenue de l'Europe 78140 FRANCE

Abstract

Since 1973, C.I.I. has been developing a high-power computer architecture built around approximately 100 interconnected microprocessors. Special hardware and software studies have been conducted to maximize processing throughput, and an architecture simulation system has been built (1).

This paper discusses a high safety architecture derived from the one above. Special emphasis has been placed on multi-microprocessor applicability of the MARIGNAN Method (2) which features

- all-processing dual-redundancy,
- dual processing result comparison for result verification,
- sequence restart if failure (s) detected.

Permanent failures entail automatic faulty microprocessor elimination and, in turn, automatic rollback of interrupted processes.

This proposed architecture offers the following advantages .

- high safety-to-cost ratio,
- hardware/ software compatibility with standard multiprocessor architecture,
- software transparency.

Introduction

LSI technology developments have led to the appearance of microprocessors on the market that are being used to build low-priced, low-throughput computers in a few packages. Current evolution of these microprocessors tends to increase power and reduce prices, but foreseeable performances of the microprocessors are not envisaged to attain those of current large-scale computers at reasonable costs.

As a result of this C.I.I. has undertaken structure research based on the association of a large number of microprocessors to attain the power of current large-scale computers The structural aspects of this research could be feasible during the first half of the 1980s.

Foreseeable user needs concerning safety and availability in the 1980s will be much more important than now, in particular, in the fields most apt to be developed : data base management, transaction management...

This paper discusses how the multi-microprocessor structure may achieve high safety at minimal costs

Overview of the non-redundant multi-microprocessor architecture

Main Multi-Microprocessor Architectural Characteristics will be discussed. For more details, refer to the various technical reports of the project.

Hardware structure

Assume that we would like to build a large-scale computer for data processing center applications (e.g., data base management, transaction management) with microprocessors. Estimating throughput of each microprocessor to be 1/ 10 of the large- scale computer throughput and architectural efficiency to be approximately 1/ 10, the required microprocessor number will be approximately 100. So that the computer has features comparable to those of current large-scale computers, these microprocessors should share a high-capacity main memory (several megabytes) slow enough to be cost-effective. All these characteristics lead to the following architecture :

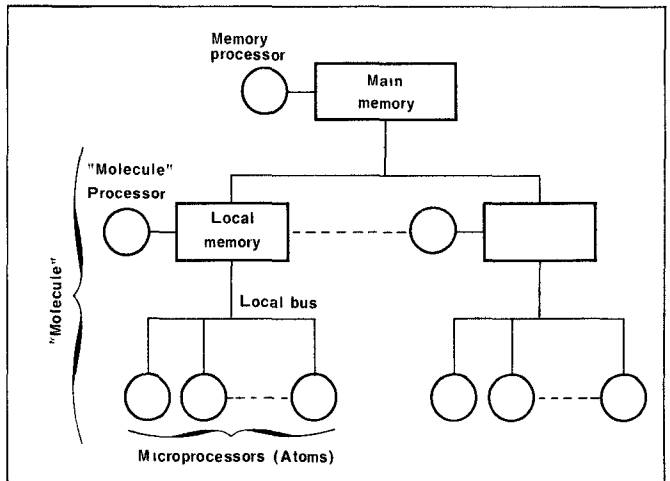


FIG 1 - NON-REDUNDANT MULTI-MICROPROCESSOR ARCHITECTURE

The microprocessors (called "atoms") are grouped into "molecules". Each "molecule" comprises

- a few atoms (on the order of 4 to 8),
- a local memory,
- a "molecule" processor.

Molecules access main memory via a memory bus. Main memory is managed by a memory processor.

"Atoms"

Atoms are microprocessors all with the same hardware but specialized firmware to execute one of the following functions .

- execution of CII X-coded instructions (CII machine code close to the IBM 360 code),
- execution of high-level-language programs (COBOL, ALGOL, FORTRAN...),
- data base management,
- peripheral management
  - disks
  - printers
  - transmission lines
  - etc...

Each molecule can contain only atoms of the same type.

Molecule processor

The molecule processor is used to manage exchanges between main memory and local memory and exchanges between atoms and local memory. Moreover, it manages fine parallelism (see paragraph *"Fine parallelism"*).

Local memory

The local memory comprises an associative cache buffer (CB) storage and memory areas used for fine parallelism management (see paragraph *"Fine parallelism"*).

The programmer-transparent cache buffer storage is used more to reduce the number of accesses to main memory than to reduce access time from the atoms.

Information stored (both in main memory and in the cache buffer storage) consists of three types :

- program instructions . these are shareable between processes, but cannot be modified,
- global data · these are data shareable between processes, but under associated resource protection via instructions TES (TEst and Set) and RESET,
- local data . they belong only to a single process.

Operating system organization

Any processing is divided into "processes" managed by conventional FORK and JOIN primitives.

The process-to-processor allocation mechanism (the "hypervisor") is based on the concept of a "Logic Processor" which groups processing ' capacities of the same type All physical microprocessors of the same type (emulators of a given code or input/ output processors that share multiple-access peripherals, etc.) constitute a logic processor with an associated queue. A process that shall be executed on a logic processor will be, therefore, either active on one of the microprocessor associated with this logic processor or waiting if the number of active processes on the logic processor equals the number of microprocessors

When a process is placed in the queue of a logic processor, a signal ("bell") is sent to all the physical microprocessors Inactive microprocessors consult the queue of the logic processor to which they are associated. If the queue is empty, the microprocessor does nothing. If it is not empty, it acquires the first process of the queue and tests if all the queues are empty , if so, "bell" goes low.

"Bell" is not the only means for interprocessor dialoguing. In fact, a process should be capable of being halted, e.g. in case of looping. To do this, a HALT signal or "ANTI-BELL" signal is tested periodically by the active microprocessors If HALT is high, the microprocessor will consult a queue of processes to be halted and compare the name of the process that is being executed If identical, the microprocessor aborts its process and goes inactive.

Microprocessors are not identified (except by the process name that they execute). Therefore, they are not addressable. All microprocessors associated with a logic processor are non-specialized. Any inter-process data communication passes, therefore, through main memory. Moreover, a process is created or destroyed by processing queues in main memory (FORK, JOIN) , that is thus costly. Consequently, it is desirable to define a finer parallelism.

Fine parallelism

It entails scattering a process into several branches ("mini-process") in as flexible a way as possible for execution on the different microprocessors of the same molecule. This offers the advantage of having easy communication between branches via the cache buffer storage (CB). Moreover, cache buffer storage utilization is optimized since the various branches work on the same data copy in the cache buffer storage.

A miniprocess management which is lighter than process management has to be envisaged, and, in particular, queue processings should be avoided. Thus the tree structure generated by the mini-forks and mini-joins will be stored by counters and pointers at the local memory level of the molecule and managed by the molecule processor which could also be responsible for cache buffer storage management Like BELL and ANTI-BELL for process management, a MINI-BELL and a MINI-ANTI-BELL exists at the molecule level for miniprocess management

High safety structure constraints and objectives

Safety rather than availability or reliability

First, these terms will be defined within the multi-processor context :

- reliability at the component level is the probability that at any given time the component has not failed At the system level, reliability is the probability that the system satisfies its "mission". This notion of "mission" is perfectly defined in time and in processing for certain special systems such as e g. systems on-board space vehicles. But the concept is less precise for a data processing center system
- availability is generally defined as the ratio of system operational time to total time

$$\frac{MTBF}{MTBF + MTTR}$$

In a multi-processor system, it is more natural to talk about "availability ratio" defined as the proportion of the system which is in an operational state.

- safety is not universally defined in data processing. In fact, it is linked to the "seriousness" of the failures which can be defined only for an application known with precision From a first approximation, however, any failure will be considered "serious" that causes an error that might be propagated outside the system, i e. on input/ output devices, e g. by transmitting an incorrect message on a terminal, or by writing a false record on a file, etc Serious failures are distinguished from "fatal failures" which halt the system Based on these definitions, a serious failure may be not fatal an a fatal failure may be not serious Now safety will be defined as the probability that no serious failure occurs in a time unit. Reliability will be rather the probality that no fatal failure occurs in a given time.

For a multi-microprocessor, processor non-specialization and structure modularity make a fatal failure improbable provided certain critical points have been studied carefully to insure high reliability . power-supply, memory bus, etc.

Outside these critical points, reliability is not a major problem, because the failure of one of numerous identical units will just lead to a graceful degradation.

Moreover, the number of physical processors of the multi-microprocessors is around 100 But first estimates lead us to believe that it will be difficult to obtain such a high parallelism. There will, therefore, be a number of inactive processors Hence - for a repairable "data processing center" type system - availability will only be a secondary problem It is easy to schedule sufficient maintenance with respect to component reliability so that the number of failed processors does not significantly penalize overall system throughput

On the other hand, system safety is a major objective In fact, high parallelism may provoke error propagation faster and more extensively than in a single-processor system thereby making rollback and failure localization procedures more difficult In addition, considered from the viewpoint of a large number of resource-sharing users (rather than time-sharing users), file recovery or input/ output correction procedures may become particularly arduous for both the system designer and user

Cost/ safety tradeoff

Safety, of course, is not free ; it requires error detection and correction or error masking devices that will be costly in time and/ or hardware , these devices would not be really necessary if throughput were the only objective,

Two extreme positions - both unacceptable - are ·

- a system with zero throughput; it will, of course, be completely sure, but will not be cost-effective,
- a system without any safety device, it will, no doubt, have an interesting throughput, but results will be questionable

To optimize these opposed criteria, one should be expressed in terms of the other ; e.g estimate cost of a serious failure and deduct the structure at minimum global cost. This type of estimate could be employed for certain precise realizations. But for a general-purpose system, considering the large number of different applications, it is difficult to estimate the cost, e.g. if a file is destroyed.

Likewise, in the current state of this multi-microprocessor study, the cost of each unit cannot be set precisely

Consequently, a more intuitive approach will have to be used. Our search will be to find satisfactory safety for a reasonable cost.

Compatibility with non-redundant structure

C I.I. Multi-microprocessor Structure research has been significantly developed based on hardware/ software aspects with performance rather than safety as the prime criterion.

To take full advantage of the most interesting original concepts of this research, it is desirable that the high safety structure be compatible with the non-redundant structure in terms of both software and hardware.

Hence, it will be necessary .

- to insure redundant structure transparency at the software level,
- to minimize hardware modifications.

Moreover, it is conceivable in the foreseeable future that both structures will coexist ·

- i.e , a cost-effective structure with conventional test, reconfiguration, and rollback means for data processing center system oriented toward batch processing,
- a high safety structure for systems under extremely severe requirements . real-time systems, highly interactive systems, etc.

Under these conditions, two structure compatibility is desirable to reduce study, manufacturing, and operational costs.

Proposed redundant structure

Initial assumption

Intermittent failure detection

Based on our definition of safety, a certain number of requirements will now be deduced

Different studies (4, 5) have shown that 80 to 90 % of the errors arising in data processing non-redundant systems are due to intermittent failures Since these errors are quite capable of affecting input/ output, these errors are "serious", and should be detected. Therefore, it is not sufficient to periodically check system operation (programmed or microprogrammed tests) ; processed information validity as well as processing validity have to be continuously monitored

Redundant codes

But the multi-microprocessor structure does not permit easy continuous monitoring of information with low redundancy (arithmetic codes on arithmetic operators, duplication for the logic operators, detecting or correcting codes on data transfers ,and commands, instruction retry.) because this would require major modifications of microprocessors which would not be compatible with the non-redundant structure and which could not be easily built with standard LSI packages.

In addition, redundant code safety is limited by different factors .

- command monitoring difficulties,
- arithmetic operation verification difficulties (arithmetic codes) and logic operation verification difficulties (duplication is the least costly code for logic operations),
- detector fragility

Therefore, these codes cannot be used for processors. On the other hand, correction codes can be used for memories and buses due to their low overhead and their efficiency in these fields

Modular redundancy

Modular redundancy is preferred at least for processors It entails performing the same processing several times on the same unit (time redundancy) or on several different units (space redundancy), and then comparing the results

Time redundancy is a priori rejected because it does not permit detecting permanent failures (which are also capable of affecting input/ outputs), and because it is also as costly for multi-processor availability as space redundancy.

And the redundancy level now has to be selected. Several levels are possible between the component and the global system. For the sake of simplicity and compatibility with the non-redundant structure, it seems worthwhile to add redundancy to the molecule level. Each processing will be performed on several molecules and results will be compared.

It is now necessary to specify the degree of redundancy :

- dual redundancy permits detection of all simple failures and nearly all multiple failures that have produced an error on processing- it is therefore extremely "safe" , but error correction must be insured by other methods , breakpoints, failing processor identification
- a higher redundancy that permits masking the error by a majority vote, but this solution is at least 1.5 times more costly than the previous one.

The most reasonable structure in terms of cost and safety that can be intuitively defined is therefore a dual space modular redundancy at the molecule level

The results from one of our previous studies (2) can now be recalled and adapted to the context of the multi-microprocessor structure.

The Marignan method principle

The method selected consists of executing the same processing on two processors ; these two processors must wait for one another at the end of each "sequence" to exchange check-sums computed on the results of their processing , if both find check-sums equal, they execute the next sequence, otherwise, they return to the beginning of the sequence.

The "sequence" is defined as being the largest portion of the execution of a task in which - if a failure occurs - it is sure that an error cannot be propagated outside the task In practice, the sequence is the sequencing of a task between events liable to propagate the error such as :

- global data modification,
- resource release/ reservation,
- process creation,
- input/ output operation

Let us see how these principles can be applied to the multi-microprocessor structure

Pairing

Molecules are statically paired any processor of one molecule is bonded to the corresponding processor of the other

This bond will be discussed below (paragraph *"Resumption - Suicide - Fratricide*). Processors (respectively, molecules) paired are called brothers (or sisters)

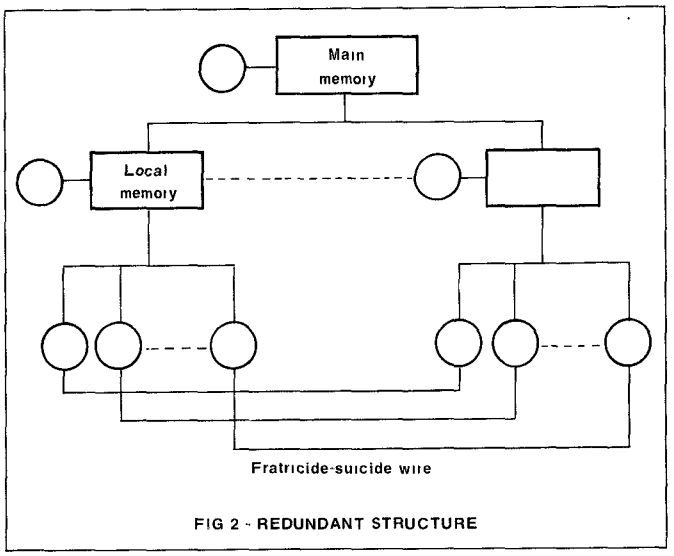


FIG 2 - REDUNDANT STRUCTURE

Static pairing offers the following advantages :

- greater inter-brother bond simplicity,
- higher suicide reliability (refer to *"Resumption - Suicide - Fratricide*),
- no processor addressing.

The disadvantage of a lower availability in case of several processors failing can be neglected in the context of a data processing center system where maintenance is possible before a large number of processors are out of service

Note that this dual redundancy at the molecule level permits checking the processings executed by the atomic processors and by the molecular processors as well as local memory cache buffer operation

Notion of sequence , comparison , deferred copy

Any process accepted by a processor is also accepted by the "brother processor".

"End of sequence" event

Both processors perform the same processings up to the next "end of sequence". Practically, the event causing an end of sequence is always a TES or a RESET. In fact, TES and RESET protect :

- system table access,
- resource reservation or release,
- global data reservation,
- process creation and end

The mini-process management system is simply protected by reserving the non-shareable resource that is the molecule processor Since this reservation remains within the molecule, it cannot cause an "end of sequence".

Deferred copy

During a sequence, any store in memory will be limited to the cache buffer storage. Main memory will be modified only at the "end of sequence" this is the deferred copy for which solutions of the non-redundant structure deferred copy method shall be used

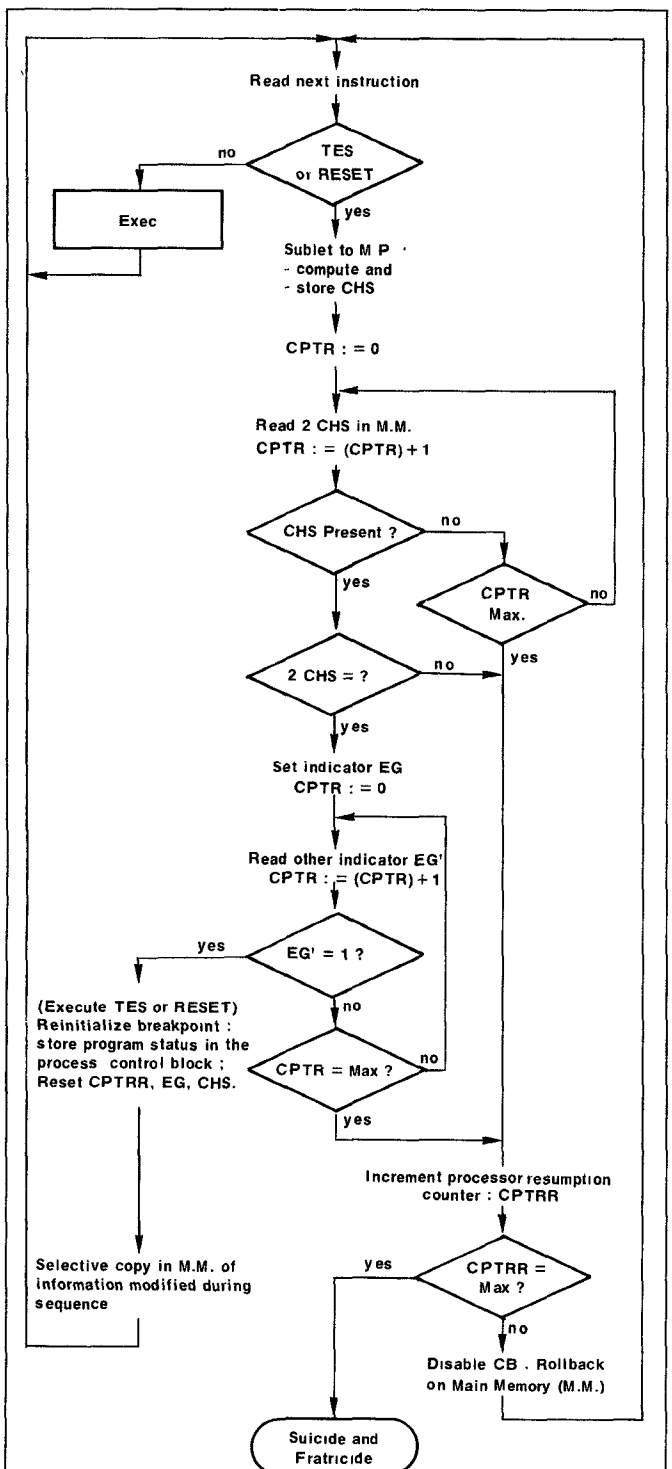
Since data from a process in main memory is not modified during a sequence, they can be used as a "snapshot" for possible rollback to the beginning of a sequence. Consequently, checkpointing is very cheap.

"End of sequence" sequencing (see flow chart 1)

At end of sequence, the molecule processor computes the check-sum CHS of data modified during a sequence by the processor reaching its end of sequence , then the molecule processor will store the computed check-sum in the process control block. Both processors at end of sequence loop on the read of both check-sums. When a processor has read both check-sums, it compares them ; if it finds them equal, it sets indicator EG in the control block, and loops until its brother has also set its indicator EG'. When this is done, it reinitializes the breakpoint (save program status in the process' control block . ordinal counter, indicators, registers...) executes (alone) the deferred copy. and both processors go on with the execution (next sequence).

Resumption - suicide - fratricide

If a synchronization has not been made (exchange of CHS or exchange of equality indicators EG), or if both check-sums are different, a failure is deduced to have occurred during the sequence. Therefore, it is necessary to resume execution at the beginning of the sequence by restoring the last program status and by disabling the process-modified blocks in the cache buffer storage. However, if the failure is permanent, all rollbacks will fail, and the two brother processors will have to be placed off-line from the rest of the structure to reinitiate the process elsewhere.



FLOW CHART 1 : END OF SEQUENCE

To do this, a rollback counter CPTRR incremented at each ROLLBACK is associated with each processor, and reset at each end of sequence without error. If the rollback counter exceeds a maximum value, the error is declared permanent and both processors must go off-line. To do this, they are bonded together by a "fratricide-suicide" wire. It is sufficient that one of the two brothers acts on this wire for both processors to be isolated from the system , this wire is the minimum link between the two brother-processors. The process which was active on these two processors will be reinitiated by the watchdog timer.

Remarks

Reliability - safety:

The fratricide - suicide wire causes a reduction in reliability ; it is sufficient that one processor by mistake (even intermittently) activate this wire for both processors to be lost for the system. On the other hand, this solution provides better safety , if a processor is permanently failing, its brother kills it and itself before the failure can be propagated as an error outside the sequence. Note that the probability of a processor with an intermittent failure randomly activating the suicide wire is extremely low since the corresponding micro-instruction is rare. Reliability of this system is therefore correct.

Molecule processor and memories safety:

This redundancy method doubles the processing of the "atom" processors but also the molecule processors and the local memory. Hence their proper operation is indirectly checked In case of permanent failure on a molecule processor, the molecule does not commit "suicide" immediately. But all processors of the molecule will successively commit suicide. The molecule will thus be "isolated" from the system. Consequently, in the redundant structure, the following will not be doubled :

- the memory bus
- main memory and auxiliary memory,
- the memory processor.

The memory bus, main memory, and auxiliary memory are especially protected by using a correction code on the information (and, if necessary, on the addresses). The memory processor role is to act as an interface between main memory and the memory bus (and, if necessary, to manage the transformation of a virtual address into a real address, if the cache buffer storage "works" in virtual address which makes access faster facilitating address transformation table management). Safety is insured by realizing it in the form of two identical computers either microsynchronized with a comparison by special hardware, or synchronized with each output to the memory bus or to main memory with result comparisons. Rollbacks are facilited by the non-volatile feature of the inputs : main memory or buffer of the interface with the memory bus.

Input/output:

Two cases have to be studied ; that of auxiliary memories and that of other peripherals.

For auxiliary memories, it is generally possible to double the peripherals. This is the simplest, conventional solution in certain transaction management or data base management systems with high safety.

Input/output processors are doubled in this case leaving them asynchronous for outputs but with a check-sum computation performed in the local memory buffers for inputs and a check-sum comparison.

For other peripherals, which normally are not doubled, both input/output processors will be compared before each output. This comparison will, therefore, be performed by special hardware. In input, the same information from the peripheral is sent to both processors and is buffered in local memory where two check-sums are computed and compared before one of the processors makes a copy in main memory. Under these conditions, neither the peripheral nor the peripheral-comparison hardware link are checked since they cannot be doubled.

Safety consequences

The processing of each atom processor is checked for each sequence. Likewise, the consequences of processing molecule processors on sequences evolution, on check-sum computations, and on memory access management are checked. Check-sum computation at end of sequence on data modified in the cache buffer storage permits checking the validity of information that will be copied in main memory. Cache buffer storage failures liable to contaminate main memory are therefore also detected. Even if data exist in a single copy in main memory, there is a guard against most of the bus and storage failures encountered by using a correction code Good input/output safety can be insured by the mechanisms proposed in *"Input/Output"*.

Cost consequences

For processing, two molecules are required where only one was sufficient in the non-redundant structure. For a given number of molecules, apparent availability is thus half of that of the non-redundant structure.

This should not reduce machine throughput to half since it is probable that it will be difficult to obtain a parallelism that suffices to use at the same time all microprocessors, and therefore that a certain number of micro processors will be inactive

Reciprocally, to keep the same throughput, twice as many microprocessors will have to be in the redundant structure as in the non-redundant structure. But this should not cost twice as much due to system modularity and element standardization.

Moreover, in a machine such as the multi-microprocessor, it is probable that main memory will represent a major portion of the global cost. And this memory is not doubled ; a correction code is simply added and the data path widths will be increased approximately 15% (main memory and memory bus).

Other special architecture features

A certain number of other problems have been resolved, but their development is too long to be discussed here.

There are .

- process acceptance - realized by a brother microprocessor to brother microprocessor dialogue,
- cache buffer storage management
  - structure
  - deferred copy
  - consistency between the various caches
  - *read direct (RD) in main memory*
  - saturation
- watchdog timer - realized by permanent processes,
- resource recovery in case of a permanent failure - realized by watchdog processes scanning the resource tables.

Conclusion

The feasibility study has shown that the envisaged architecture responds to the objectives that we imposed, viz. :

- safety,
- reasonable cost (processor duplication and inexpensive checkpoint usage),
- compatibility.

A bread board model of this architecture is underway to evaluate with greater precision the safety, cost, and throughput parameters related to the non-redundant architecture

Acknowledgment

The author is deeply grateful to the team that studied the Multi-Microprocessor Architecture, and, in particular, to Mrs. Alice RECOQUE, Miss Dominique BORRIONE, Mr. Louis CENSIER, and Mr. Guy MAZARE

This study was supported in part by the "Direction de la Recherche et des Moyens d'Essais" (D R M.E) under contract no 73,375.

Referenes

1 - D. BORRIONE .

"Lascar · A Language for Simulation of Computer Architecture" 1975,International Symposium on Computer Hardware Description Languages and their Applications New-York September 1975

2 - Y. DESWARTE & J. LAVICTOIRE

"MARIGNAN . An Intermittent Failure Correction Method" FTC 5 Paris June 1975.

3 - A. RECOQUE :

Journées d'Etudes IRIA St Pierre de Chartreuse - 22/23 Novembre 1973

4 - M. BALL & F. HARDIE

"Effects and Detection of Intermittent Failures in Digital Systems" AFIPS, FJCC 1969

5 - G. H. MAESTRI

"The Retryable Processor" AFIPS, FJCC 1972

Yves DESWARTE was born September 24, 1949, at ROUBAIX, (FRANCE). He recieved the degree of Ingénieur from the Institut Supérieur d'Electronique du Nord (ISEN) in 1972, and the degree of Computer Science Specialist from the Ecole Nationale Supérieur de l'Aéronautique et de l'Espace (ENSAE) in 1973.

In September, 1973, Mr. DESWARTE joined the Military, Space, and Aeronautics Division of the Compagnie Internationale pour l'Informatique (CII) where he has been working on research contracts dealing with data processing systems safety, reliability, and availability.