A Fault-tolerant Real-time Microcontroller with Multiprocessor Architecture

Elio Strollo
Department of Electronic Engineering
University of Rome "La Sapienza"
IES Ingegneria Elettronica Sistemi S.r.l.
Rome, Italy
elio.strollo@iessrl.it

Alessandro Trifiletti
Department of Electronic Engineering
University of Rome "La Sapienza"
Rome, Italy
trifiletti@die.uniroma1.it

***Abstract*—The occurrence of faults within microelectronic systems concern an increasing number of application fields. A possible way to build reliable systems is make them fault tolerant by the adoption of redundancies. In this paper we present the design and implementation of a fault tolerant microcontroller device using hardware redundancy of a multiprocessor. This system is FPGA based and parametrized, and can be configured to operate both as a multiprocessor or a single redundant processor, with three processors working in TMR and some spare processors. The indicated recovery procedure allows to rapidly correct transient faults, but also to replace the damaged processors after a permanent fault. The complete system employs three redundant processors on distinct boards, that are used in conjunction so as to have a further TMR level. In this way we obtain a module that is capable to solve all of the system faults, both at processor or boards levels. The programmable system is capable of masking faults, correcting errors, and also showing the possible gradual system degradation due to permanent faults.**

***Keywords*—Fault tolerance; reliability; multiprocessor; hardware redundancy; reconfigurable soft-core; failure recovery.**

I. INTRODUCTION

The microelectronic systems for aerospace and avionic applications work in environments where faults, due to radiations or high energy particles, that interact with the circuits, may happen. Now the failures and reliability issues affect the chips of current and future generations also used on earth.

The dimensional scaling in the manufacture of VLSI integrated circuits brings to small capacitive and voltage values. This leads to an increasing sensitivity and vulnerability to radiations, particles, electromagnetic interferences and aging. Besides this, also the fabrication flaws increase due to the higher process variations, and to faults for electromigration and other phenomena, so the circuits become more and more unreliable.

A possible way to address circuits faults within a processor, both due to the aerospace use and to the scaling, is to implement fault tolerant architectures.

II. BACKGROUND

Briefly, we consider the scenarios and the problems. The radiation in space consist of particles and photons which, impacting the material, determine energy transfer processes, dislocations and ionizations. On the earth, the shielding effect of the atmosphere and of the terrestrial magnetic field reduces the energy of these particles.

The main effects on the circuits are classified in Single Event Effects - SEE, comprising

- Single Event Transient – SET (voltage pulse/spike due to momentary power/lighting of a transistor),
- Single Event Upset – SEU (inversion of the logical value in a memory cell),
- Single Event Latch-up – SEL (activation of the parasite substrate transistor, this event is often destructive),
- Single Event Burnout – SEB / Single Event Gate Rupture – SEGR (destructive events),

or Total Ionizing Dose Effects – TID (due to the charge accumulations).

TID effects, which, for example, changes the threshold voltage of gates, contribute to the deterioration of the device over time.

Thus these ionizing radiation can cause soft-error, such as the inversion of a bit in a register, and other recoverable failures, which are transient faults, or can cause permanent or intermittent faults.

Even in non-space environment, the current use of nanometric geometries in the manufacture of integrated circuits leads to reliability problems. The reasons are:

- for probabilistic nature, the large number of transistors leads to the appearance of defects into the chip and thus to the impairment of the device;
- the small size imply reduced/smaller capacity and voltages, this therefore leads to higher/greater sensitivity to external and internal disturbances (radiation, electromagnetic interference, noise) and to parameters variation and temperature;
- process variations are much greater and this entails increased diversity between the transistors;
- become even more relevant failures for electromigration and aging phenomena.

Thereby increases the defects during the manufacture, reducing the yield, and increases faults during the operating life of the devices.

ISBN 978-83-63578-08-4

Regarding the radiation, they are designed solutions, at the level of transistors and circuits, to make these more robust and able to withstand high incidences of ionizing particles. These techniques include the use of larger transistors, the increase of the operating voltage, the gate oxide reduction, the addition of further transistors.

However the use of Rad-Hard circuits has a considerable cost, because only some foundries have the related manufacturing processes, for the limited demand, for the higher intrinsic cost. Moreover the use of these circuits involve lower performance.

An example of Rad-Hard processor is the BAE RAD750 based on the PowerPC 750 architecture. Its clock frequency is 200 MHz, while the default processor reaches 1 GHz. Furthermore, the cost of RAD750 is three orders of magnitude greater of the PowerPC 750.

Fault tolerance techniques have been proposed, and are used, both for space systems, and also as a possible solution to increase the yield of VLSI systems and cope with the unreliability of ultra-scaled nanotechnology.

III. Fault Tolerance

Fault tolerance techniques have been conceived for a long time, the first studies date back to von Neumann. There is now a renewed interest due to the new demands.

The fault tolerance is obtained by the introduction of redundancy, these redundancies may be in the hardware, in the software in the time, in the data. An example of redundancy in the data (information redundancy) is the use of error correcting codes (ECC) in the memories. An example of redundancy over time (temporal redundancy) is an operation repetition. An example of software redundancy is the comparison of two different algorithms for calculation. An example of hardware redundancy is the use of two copies of a critical component for detecting an internal error.

Hardware redundancy can be introduced at different levels of the project,

- at circuits level,
- at component level or
- at system level.

A typical technique is the TMR (Triple Modular Redundancy), in which the outputs of three identical modules is selected, according a majority criterion, by a dedicated component ("voter"). This allows you to detect the error and mask it and still get the correct result.

For example, the Boeing 777 uses a system in which the results of 3 different processors (Intel 80486, Motorola 68040, AMD 29050) are compared, the module is tripled for have an additional TMR level.

IV. Goal

The context is that of embedded real-time systems for space or mission-critical or safety-critical applications. The objective is to provide a programmable digital device highly reliable. The device must tolerate transient and permanent faults due to any cause.

The device has some inputs and some outputs to perform its generic control functions, and it will be based on processing elements that run a stored program code. It must be able to show their integrity state or a reduced reliability and make the transition into a safe state when it can not guarantee flawless operation.

The control device, in addition to spatial or avionics applications, or to critical applications, can be intended for applications with long times of mission or not serviceable/maintainable applications.

V. Methodology

Multiprocessor architectures are getting popular and the paradigm of the future computer systems will be inevitably parallel type.

So the availability of multiple processors can be exploited, as well as to increase performance, also for the realization of fault-tolerant systems.

We plan to leverage the hardware redundancy of a multiprocessor architecture, doing work together three processors at TMR configuration and using the remaining processors as spare parts to tolerate the permanent faults.

We design the processor with the above goals, developing the VHDL code of the overall architecture, and we implement the device in FPGA, which can be tested. For this system, we develop a probabilistic model with Markov processes.

The approach of introducing hardware redundancy at the component and system level is in order to operate with the VHDL within the programmable logic. In a possible integrated realization of a similar device, circuit protection strategies can also be introduced. As well as the design of this fault-tolerant processor can also be used in Rad-Hard FPGA.

In addition to mask the faults in the processors by redundancy, we want to activate a procedure to correct the failures occurred and to recover the system in a more efficient way of the simple shutdown and restart.

Historically, various schemes have been designed in which are used in various ways redundant modules.

Some possible approaches, that we can imagine for a multiprocessor, are:

1) TMR of the processing modules;
2) TMR of the processors with fault detection and shutdown of the discordant processor and another, moving from 3 to 1 processor (TMR with reduction);
3) TMR of the processors with detecting and masking of the first fault and then power off of the discordant processor, subsequently next fault between the two remaining processors will switch the system to a safe state, i.e. switching from TMR configuration (3 processors with fault masking) to copy configuration (2 processors with only fault detection);

4) TMR of the processors with recovery for transient errors (by realignment procedure);
5) TMR of the processors with replacing faulty processors with spare (hot or cold) and recovery/restore (by realignment procedure);
6) NMR (N-Modular Redundancy), with odd N;
7) NMR reconfigurable through the removal of faulty processors (self-purging).

The TMR criterion often is applied to processors that are considered integrally as modules. Here we choose the approach 5 and we consider separately the program memory and RAM work memory because this gives us a better recovery action as will be shown.

Considering faults at level of any chip circuit (think, for example, to the PLL or the I/O of the FPGA) or also of the board, we realize and combine the output of 3 boards, each with FPGA in which a redundant processor is implemented. The TMR of the boards is an additional level of redundancy, so we assume to combine multiple security strategies to achieve full coverage of possible faults in the system. And we compare the reliability graphs.

VI. Multiprocessor Model

The project is derived from that of a multiprocessor microcontroller for hard-real-time applications, which we have implemented in FPGA. This multiprocessor has the block diagram shown in fig. 1, each processor has a memory from which it reads the instructions program. This may be a RAM and, in this case, it is loaded by withdrawing the program from a non-volatile memory type EPROM or FRAM (Ferroelectric RAM, inherently radiation resistant). The processors read and write the RAM work memory that is split into a private part and a common part, in this latter, the locations are accessible from any processor for both reading and writing. The I/O is mapped in shared memory and to this are connected shared resource as an FPU and an external memory.

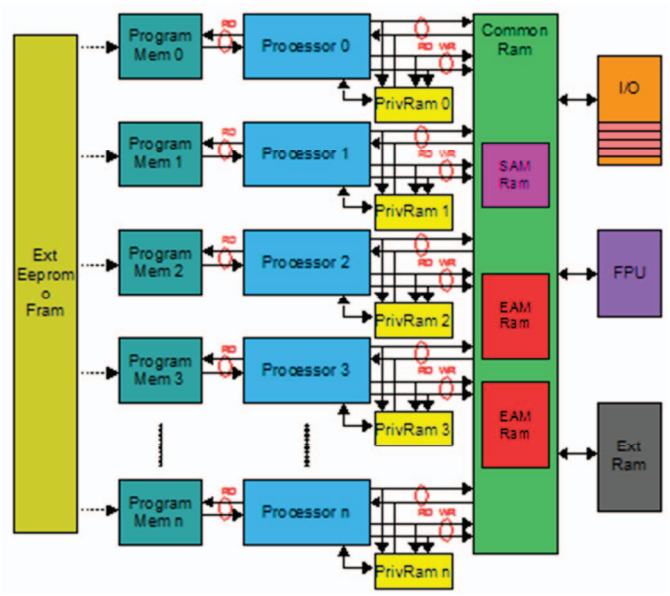

Fig. 1. Multiprocessor block diagram.

To make the shared memory practically feasible in FPGA, we have limited the number of common locations (Simultaneous Access Multiport RAM), used for example in synchronizations between processors, and added switchable RAM banks (Exclusive Access Multiport RAM) that can be connected to any processor for reading and writing independently.

The processing is completely deterministic because there is no uncertainty in communications between processors. Each instruction is executed in a single clock cycle, wherein the next instruction is read from the program memory and takes place simultaneously reading and then writing the data memory.

A representation of processing unit is shown in fig. 2. Address and instruction lines (PMADDR, ISTR) with program memory, data and address lines for read (RADDR, RDATA) and write (WADDR, WDATA) with RAM work memory, are present.

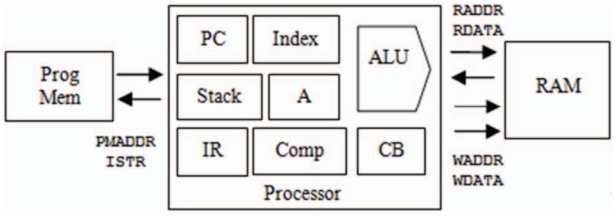

Fig. 2. Components of the processing unit.

The processor has a full ALU for calculations and an accumulator is used (A) to perform the operations. A specific component (CB) is used to manage the jumps according to read data and comparison register (COMP). Three index registers (INDEX0-2) are used in the data memory indirect addressing. In addition to the Program Counter (PC) and Instruction Register (IR), it has a Stack for the return addresses from procedures.

VII. Redundant Processor

Let's assume that you run the same code on each processor. We can introduce the voter only on the outputs of the data memory. Possible errors in these memories or errors in the data written by a processor are resolved by the voter. A 1-bit voter, to the level of logic gates, is the known circuit of fig. 3. The voter can also detect the discord of any of the inputs, just add the xor gates as in fig. 4. The disagreement of a processor can activate the procedure for recovery from transient errors.

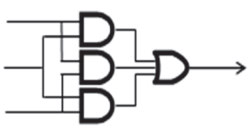

Fig. 3. Voter (1-bit).

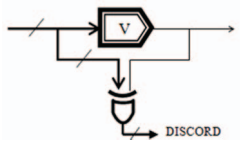

Fig. 4. Disaccord signals.

However with only a voter on the output of the main memory it is not possible to distinguish where and when the failure occurred. We introduce voter on address and data lines of all memories, as shown in fig.5, but imagine a voter on each line type. Thus there are 6 voter, each receives inputs from 5 modules and its output goes to 5 modules. This allows you to detect where the fault is generated, whether in program memory and in data memory or within the processor. So take action for focused and fast correction.

The voter on write address of the data memory (WADDR) check which location you write (protection of where you write), the voter on the input data to the data memory (WDATA) check the correctness of the data written (protection of what you write), the voter on read address of the data memory (RADDR) check from which location we read (protection of where you read), the voter on the output data from the data memory (RDATA) check the correctness of the data read (protection of what you read), the voter on address of program memory (PMADDR) check which instruction is taken (sequence protection), the voter on output of the program memory (ISTR) check the correctness of instruction (instruction protection).

The voter are critical component because no redundant and their failure affect the functionality of the processor. If we can not consider a low probability of failure, for example, for their realization strengthened at transistor level, then we will replicate as in fig. 5 (each voter has input lines from 5 modules and the output goes to a single module). This allows to bring their failure to failure of component connected to the output.

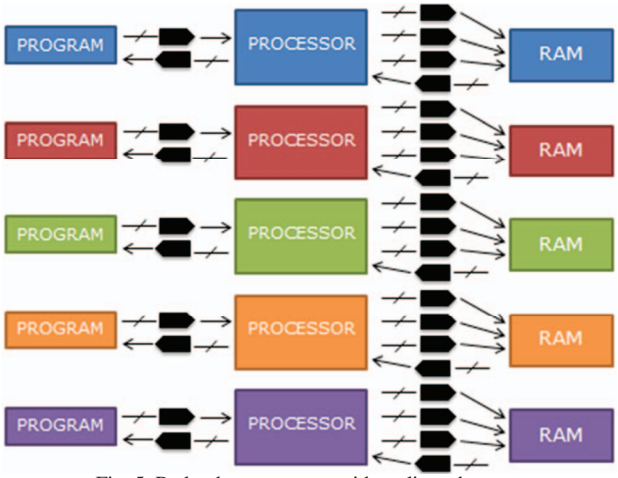

Fig. 5. Redundant processor with replicated voters.

Five processors are shown in figure, thus the voter are configurable/programmable to select only 3 inputs of 5. A possible embodiment with a multiplexer in fig. 6. Set the multiplexer on the same input can allow you to cancel the voter function to be able to do working independently the processors restoring the multiprocessor capabilities.

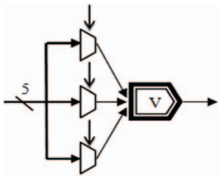

Fig. 6. Voter with selectable inputs.

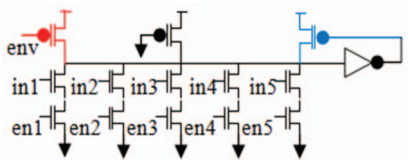

Fig. 7. Ratioed voter.

VIII. Ratioed Voter

Since the votes are introduced in the signal paths, their delay influence the clock period and the maximum frequency of the processors. Instead to realize it to logic gates level, to reduce the propagation time and the size, it is assumed a transistor level design.

The proposed circuit (fig. 7), inspired by the pseudo-NMOS logic is based on the dimensional relationships between the NMOS and PMOS (ratioed logic). It provides an NMOS for each input, in series with a transistor to enable the input. A PMOS always in conduction constitutes the pull-up network. If we enable always three inputs, the activation of one pull-down branch must not be able to bring the voltage of the drain node to low value and must be necessary to activate at least two branches. Called "i" the capability to conduct current of the two NMOS in series, the PMOS must be able to conduct approximately 1.5 "i".

It is possible to add a PMOS in parallel (by recalculating the geometries), controlled by the inverter output, to reduce the fall time at the expense of rise time (blue transistors). However the processor is meant to be implemented in dynamic logic and the charge of the voter node can take place during the precharge phase of the logic.

Obviously a circuit disadvantage is that there is a static dissipation when a branch of the pull-down is activated.

We can modify the circuit to provide the by-pass function of the voter, we subdivide the PMOS in two (by recalculating the geometries), one with grounded gate, the other (red transistor) with the gate brought to zero only when enable the voting. Therefore, if the voting is not enabled, the activation of one pull-down branch must be able to bring to low-level voltage and we'll enable only the input you want to transit.

IX. Recovery Strategies

The recovery procedure for temporary failures must align the internal state of the processor correcting this error. This is done by rewriting the corrupted registry. It is done only on those who may be involved. If the disagreement is on WADDR or RADDR or WDATA or PMADDR then the failure occurred in the processor, if it's on RDATA then the fault is in the data memory, whether it is on ISTR then the fault is in the program memory.

You can intervene immediately when an error occurs with a specific interrupt (interrupt by the voters) or to intervene in the most appropriate times to temporary suspension of real-time processing, for the execution of the recovery procedure, through periodic polling of the flag register storing the voters discords (polling of the flags). For example, the occurrence of a fault in the data memory immediately writes the only location accessed using the interruption, or using the polling we rewrite all locations used and we periodically realign the memory.

The recovery action can be further refined, therefore, in the case of our processor, if there is disagreement on RADDR or WADDR you rewrite the only index registers (INDEX0-2), if there is disagreement on WDATA you rewrite the accumulator and the carry and overflow flag (A, CY, OW), if there is disagreement on PMADDR you rewrite all of the processor memory elements (PC, STACK, a, COMP, INDEX0-2, CY, OW). If there are events SEU in program memory, in our case, we command reloading the program again from the FRAM memory.

We note that the recovery procedure operations are performed by the triplet of processors linked together and corrected by the voters, so the rewriting operations are always performed correctly.

As regards the permanent faults, we can verify that, after the rewriting, the data is read without error, or the approach is to store how many faults undergo the components and estimate whether a component is to be considered permanently malfunctioning. In this case, the entire processor is replaced, reconfiguring the voters and performing a complete realignment of the internal state.

X. Redundant Registers

To protect a register against SEU events, you can make it as in fig.8. The circuit is very effective because the tripled register is reloaded in each clock period with the correct data from voter. So it keeps correctly the data that is written due to the effect of the continuous cleaning (to overlook the faults of the voter, you can create one for each mux).

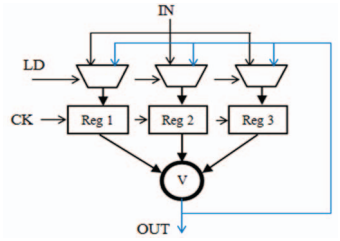

Fig.8. Tripled register.

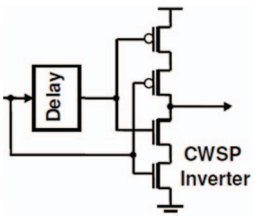

Fig.9. SET-tolerant inverter.

However the circuit increment is large, it has greater propagation times and we do not have the ability to change an eventual permanently faulty part.

In the system we use it only for the voter configuration registry (VSELECT), which is a special register. This also controls the switching on or off of the processor and switches the fault-tolerant-processor or independent-processors mode. The disagreement signals from the voter are combined and stored in another special register.

Instead, to tolerate SET events, you can filter the pulses on lines with delays and Muller C-element (fig.9).

XI. Realignment and Recovery Procedure

As mentioned, with the arrival of the discordance signaling by voter, the processor can react immediately, or it can call up the recovery process when its execution can be interrupted. We consider the second case, more oriented to hard-real-time tasks, and describe the behavior of the procedure.

In the case of only recovery of transient faults, registers and memory locations will rewrite, in the case of recovery from permanent assessed faults, it will be implemented the replacement of the processor with a spare one and to its full alignment. This will consist, after the program memory loading of spare processor, in a reading and rewriting of all active values of which assure the update through the voter correction. Id est:

- detection of the faulty processor and if it may be permanent damaged;
- if permanent, replace the processor, store that it is out of service and perform recovery;
- otherwise if temporary, run the restore based on the specific error.

Referring to our processor, in a simplified way, the behavior is described in the following. When switched on, the system is set up with the processors 1, 2 and 3 started and their program memories are loaded.

Procedure call in case of discordance indicated by a voter:

- read the number of erroneous processor and the type of discordant voter;
- update the error counter of this processor;
- if the permanent fault criterion is positive, then jump to [1];
- if divergence over RADDR, then rewrite INDEX0-2;
- if divergence over WADDR, then rewrite INDEX0-2, COMP, A;
- if divergence over WDATA, then rewrite A, OW and CY;
- if divergence over RDATA, then rewrite all active ram locations i.e. used;
- if divergence over PMADDR, rewrite INDEX0-2, COMP, A, OW, CY;
- if divergence over ISTR, then reload the processor program memory;
- return
- [1] set VSELECT register to processor replacement, store out of service and signalize processors decrease, handle eventual shift in safety mode;
- reload the program memory of new processor (waiting);
- rewrite all active ram locations i.e. in use;
- rewrite INDEX0-2, COMP, A, OW, CY, the STACK;
- return.

Therefore, in the hypothesis of time between faults much higher than response and system recovery times (accettable assumption to our use), the processor is able to overcome any transient, intermittent, permanent fault. The introduction of the voters on all lines between processors and memories, allows us to correct the transient faults quickly by few instructions i.e. in just a few clock cycles. In the case of our processor, each rewriting of register or RAM location requires one instruction and each instruction is executed in one clock cycle.

The replacement of a processor requires longer times due to the integral loading of the program memory and the writing of all the used data memory. These times will depend however on the particular application.

XII. Implementation on FPGA and Reliability

We estimated the reliability of the system via a Markov modeling, we considered a probability of failure of components dependent on the number of logical elements of the component synthesized in FPGA. In the synthesis on Actel/Microsemi ProASIC3 type FPGA, a processor (without memory) requires about 2500 VersaTiles. On A3P1000, we are 32 banks of 256 words for the memories.

In the reliability study we considered the simple “TMR” processor (Sa), the processor with “TMR + 2 cold spares” (Sb), the “TMR at boards level” (Sc) and TMR of boards containing

the processor with TMR + 2 cold spares (Sd). In fig.10 we show the diagram of the Marcov process for the system Sb (λ is the computing unit failure rate).

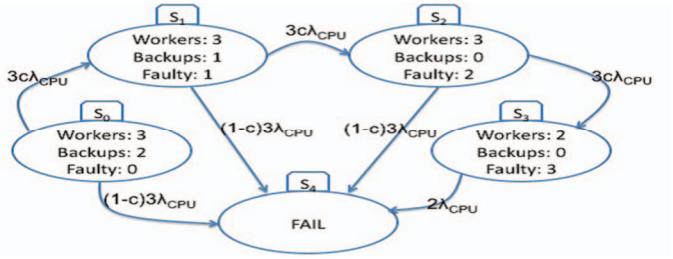

Fig. 10. Diagram of the Marcov process for system Sb (TMR + 2 spares).

With the "c" parameter we can treat multiple concurrent faults or during the recovery procedure, we set c = 1 to overlook this possibility. The equations of the model are:

$$\frac{dP_{S0}(t)}{dt} = -3\lambda P_{S0}(t) \qquad [P_{Sn} = \text{probability of system into state Sn}]$$

$$\frac{dP_{S1}(t)}{dt} = -3\lambda P_{S1}(t) + 3c\lambda P_{S0}(t)$$

$$\frac{dP_{S2}(t)}{dt} = -3\lambda P_{S2}(t) + 3c\lambda P_{S1}(t)$$

$$\frac{dP_{S3}(t)}{dt} = -2\lambda P_{S3}(t) + 3c\lambda P_{S2}(t)$$

$$\frac{dP_{S4}(t)}{dt} = 3(1-c)\lambda[P_{S0}(t) + P_{S1}(t) + P_{S2}(t)] + 2\lambda P_{S3}(t)$$

At system power we will have $P_{S0}(0) = 1$ e $P_{S1}(0) = 0$, $P_{S2}(0) = 0, P_{S3}(0) = 0,\ P_{S4}(0) = 0$. The reliability is given by the probability of not being in the state S4 ($R(t) = 1 - P_{S4}(t)$).

The graphs of the reliability features as a function of time, calculated for the different system configurations, are shown in fig.11 (non-redundant system and systems Sa, Sb, Sc, Sd) for a qualitative comparison.

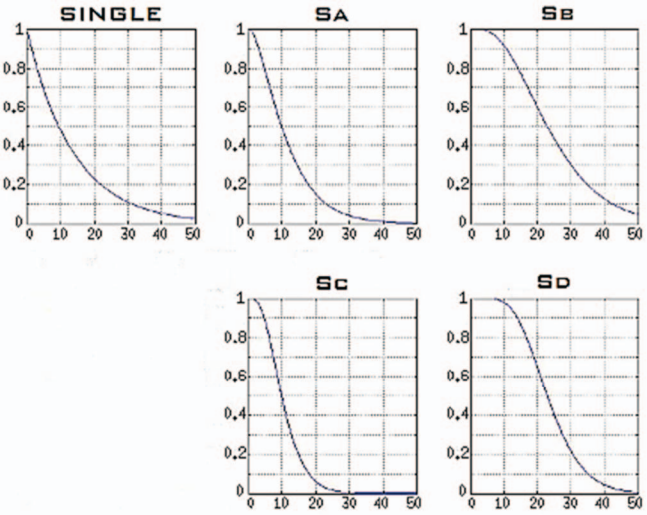

Fig. 11. Reliability of single processor, TMR processor (Sa), TMR processor + 2 spares (Sb), TMR at boards level (Sc), TMR of boards each containing a TMR processor + 2 spares (Sd).

XIII. Future Work

In the short we will carry out irradiation testing to verify in practice the system behavior under exposure to radiation of different intensities.

XIV. Conclusion

In this article we have shown one of our activity under study. The current trials will be used to size and configure the fault-tolerant system in a proper manner for future use in new business projects that require high reliability. We show the prototype in fig.12.

We consider this good approach and practically usable because you can easily choose the number of processors needed to increase system reliability and the cost of this increase within a programmable logic are acceptable in relation to the system economies. In addition, the configurability of the processors structure and of the whole multiprocessor architecture (VHDL is completely parameterized to scale the architecture as needed) allows you to adapt the system to the particular embedded application.

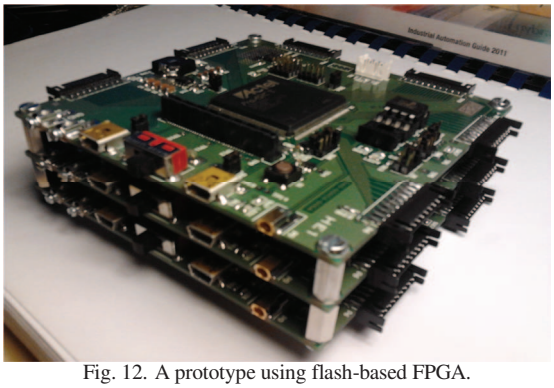

Fig. 12. A prototype using flash-based FPGA.

References

[1] Daniel J. Sorin, FaultTolerant Computer Architecture, Morgan & Claypool, 2009.

[2] Mostafa Abd-El-Barr, Design and Analysis of Reliable and Fault-Tolerant Computer Systems, Imperial College Press, 2007.

[3] Israel Koren, C. Mani Krishna, Fault-Tolerant Systems, Morgan Kaufmann, 2007.

[4] J.C. Geffroy, Gilles Motet, Design of Dependable Computing Systems, Springer Science & Business Media, 2002.

[5] Mahtab Niknahad, Using Fine Grain Approaches for Highly Reliable Design of FPGA-based Systems, KIT Scientific Publishing, 2013.

[6] Harry W. Jones, "Diverse Redundant Systems for Reliable Space Life Support", 45th International Conf. on Environmental Systems, 2015.

[7] J.Vial, A.Bosio, P.Girard, C.Landrault, S.Pravossoudovitch, A.Virazel, "Using TMR Architectures for Yield Improvement", IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, 2008.

[8] Matthew M. McCormack, Alvar Saenz-Otero, "MultiChipSat Fault-tolerant Architecture", 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, 2011.

[9] Mihalis Psarakis, Andreas Apostolakis, "Fault Tolerant FPGA Processor Based on Runtime Reconfigurable Modules", 17th IEEE European Test Symposium, 2012.

[10] Akash Anand, Rajendra Y., Shreyas Narayanan, Radhamani Pillay, "Modelling, Implementation and Testing of an Effective Fault Tolerant Multiprocessor Real-Time System", 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012.