A Fault Tolerant Hardware Based File System Manager for Solid State Mass Memory

G.C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, A. Salsano

{ cardarilli,ottavi,pontarelli,re,salsano}@ing.uniroma2.it

Department of Electronic Engineering University of Rome "Tor Vergata", Italy

Via del Politecnico 1 00133-Rome-ITALY

ABSTRACT

In this paper the hardware implementation of a file system manager for a fault tolerant Solid State Mass Memory (SSMM) is presented. A hardware implementation of the file system manager implies the following advantages: ad hoc fault tolerant design and graceful degradation capability. The former means developing special fault tolerant hardware for each file system basic function (read, write and delete). For each function different fault tolerant techniques have been applied by considering the impact of different faults on the architecture reliability. Also the area overhead introduced by the chosen fault tolerant technique has been evaluated.

Graceful degradation is obtained in terms of data connection reconfiguration and reduced functionality set. We exploited the modularity of the design to implement a distributed file system by means of local handlers on each memory module connected to a dynamic routing module. The file system manager has been used in a SSMM oriented to satellite applications.

An FPGA implementation for the complete SSMM has been obtained in order to evaluate the performances and reliability of the SSMM architecture and in particular of the file system manager.

1. . INTRODUCTION

In the past, magnetic tape recorders have been used to store the large amount of data coming from instruments loaded on satellites. While a tape recorder can provide several gigabits of non-volatile storage, its mechanical and electromechanical parts have insufficient operational flexibility and reliability for the space missions planned for the future.

On the other hand, the rapid growth in capacity of semiconductor memory devices, quadrupling every three years, now permits the development of SSMM, which are competitive with tape recorders due to their higher reliability and better performances [1] [2].

Solid-state mass memories have no moving parts and their operational flexibility has made them suitable for many applications.

The final architecture of a SSMM for space applications depends on the level of reliability and security required. Reliability is mainly related to the capability of the memory to store a minimum quantity of data after a certain working time while data security is related to assuring data integrity after hard or soft-errors. The design of a SSMM for space applications must be performed by considering reliability and data integrity requirements. Reliability improvement is obtained by introducing redundancy in the architecture; data integrity is achieved by using suitable algorithms for error detection and correction - normally based on Error Correcting Codes (ECC).

In this work system reliability is increased through the introduction of architectural redundancy. Moreover, single point of failure (typical of bus based architecture) is avoided adopting a crossbar switch matrix [3]. This solution also grants higher throughputs in the data transfer, permitting the use of parallel concurrent connections between users and memory banks. In fact, in a typical satellite application several sensors or other data collecting/transmitting devices need to access simultaneously the on board SSMM.

The SSMM logical organization is based on a file system structure. The file system is completely managed inside the SSMM and the users have no need to know the physical location of the stored data. The access to the SSMM uses file names, and high level commands for their management (read, write, delete). The hardware implementation of the file system implies different advantages: higher speed, graceful degradation through the distribution of the management of the memory modules and possibilities to use the very efficient techniques available for detection and correction of errors in hardware structures.

Moreover, these management blocks can be implemented with low hardware complexity with respect to the use of general purpose microcontrollers.

For each function different fault tolerant techniques have been applied by considering the impact of different faults on the architecture reliability. Also the area overhead introduced by the chosen fault tolerant technique has been evaluated.

The graceful degradation is therefore obtained in terms of

- Data connection reconfiguration capability (the router is able to exclude a faulty memory module reassigning the connection to another module)
- Reduced functionality set (for instance a memory module can loose its writing capability but can still read).

The paper is organized as follows. In Section 2 a brief description of the SSMM architecture is given, while in Section 3 we describe the file system design methodology. In Section 4 the hardware FPGA implementation of the proposed architecture is presented and, finally, in Section 5 the obtained performances in terms of reliability are analyzed. The conclusions are drawn in Section 6.

2. THE SSMM ARCHITECTURE

In this section a description of the SSMM is given.

The SSMM is depicted in Fig. 1. Different links can be used to communicate with the SSMM. The Spacewire (IEEE 1355 DS-DE) protocol [4] has been used in this application. In fact, Spacewire is planned to become an European Space Agency (ESA) standard for on-board data-handling in the near future and is expected to be widely used in future European missions [5][6]. Each SpaceWire link can carry data at around 100 Mbit/sec over distances of up to 10m. SpaceWire is intended to support the ready reuse of equipment developed for space applications. Moreover, the SSMM can be connected to a MIL-1553 bus, normally used in satellite platforms.

0-7803-7761-3/03/$17.00 ©2003 IEEE

The SSMM architecture can be split in several sub-units (Fig.1):

1. System Control Unit (SCU)
2. Independent Memory Array Modules (IMAM);
3. Routing Module;
4. I/O Link Interfaces;
5. I/O Memory Interfaces;

The System Control Unit manages the user access and the memory resources. This module is connected with the rest of the SSMM system through a message bus that allows communicating either the detection of a fault, or the messages necessary to handle the packet routing control.

The system uses two microcontrollers that can be connected or insulated from the system through a bypass block. Normally only a single processor is active and connected to the system, while the other one is in stand-by and electrically insulated.

A signature analysis block [7],[8] controls the correctness of the operations performed by the microcontroller.

Each IMAM is composed of several SDRAM chips, the control circuitry and a Reed-Solomon codec. The IMAM is able to support variable data word length and the EDAC unit supports variable codeword length.

The memory architecture uses *n* chips for the codeword group, *k* chips to store data, *n-k* chips to store the check symbols, and *s* chips for the cold spare. This board organization allows on-line reconfiguration of the memory module with different error correction code structures, depending on both the application considered and the data integrity requirement.[9]

In fact, it is possible to choose the codeword length among *n*, 1/2 *n* or *1/4 n*. Increasing the memory washing frequency, it is also possible to reduce the code length maintaining the overall bit error rate (BER). Of course, this solution reduces the availability. Code parameters are optimized for a specific mission using the optimization tools presented in [9][10].

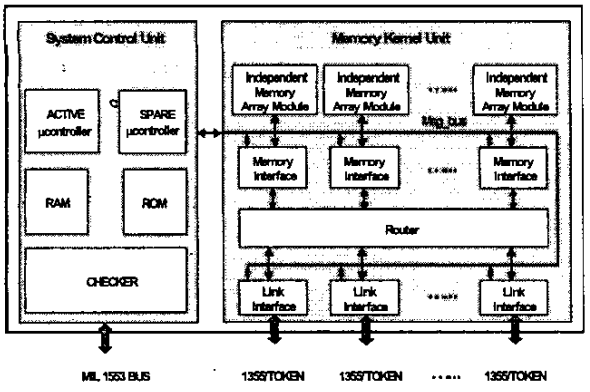

Fig.1- SSMM Architecture

I/O Interfaces are divided into two groups: I/O Link Interfaces and I/O Memory Interfaces. The packet routing control and the dynamic reconfiguration of the system in case of faults are handled by the HW/SW interaction between these interfaces and the System Control Unit. Once a connection between two interfaces is held, the data flow control is achieved by means of a full handshake.

The I/O Link Interfaces are the front end of the system providing the transport of data and messages. The Routing Module is the central switch that interconnects the users with the memories.

The paths inside the routing module are dynamically reconfigurable, as shown in [8].The memory is organized in pages of fixed length and each file is composed of a variable number of pages. The correlation between files and pages is stored in a File Allocation Table (FAT). The size of each page has been fixed to 32 KB.

Each memory interface implements the following basic functions:

- *Delete function*: used to delete a file from the FAT
- *Read function:* used to read a file from the memory
- *Write function:* used to write a file in the memory

The I/O Memory Interfaces provide the distributed file system management capability. The file system management has been designed to grant multiple concurrent accesses to the memory modules. We developed a distributed control on each memory interface that performs most of the above-mentioned functions. The System Control Unit is responsible of the dispatch of command messages to the memory interface in order to activate their built-in FAT functions. In this way, the file system management policy can be changed by reprogramming the microcontroller.

The desired reliability of the SSMM system is achieved both by means of architectural redundancies, and by introducing Error-Correcting Codes (ECC), granting data integrity (Fig. 2).

In this figure, for each unit are indicated the sub modules and in the final row, the adopted fault tolerant technique.

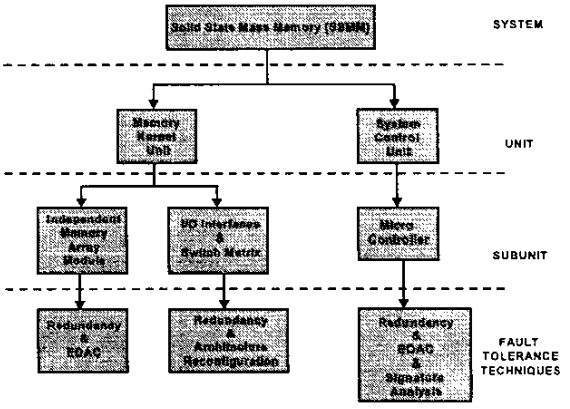

Fig. 2 SSMM partitioning and adopted fault tolerance techniques

3. FAULT TOLERANT FILE SYSTEM DESIGN

The hardware file system manager consists in a set of memory interfaces connected to the memory modules. Each memory interfaces performs independently all the FAT functions. In each memory interface the read, write and delete functions are implemented through separate HW blocks.

A system with graceful degradation capability can be developed exploiting this design approach. In fact, the SCU can exclude from the set of addressable destinations of the data traffic the interfaces affected by an unrecoverable fault. Moreover, the independent file system management on each module allows

turning off some memory modules in order to reduce both the power consumption and the failure rate.
In fact, is widely known that a de-rating factor of roughly 1/10 can be applied on the failure rate of a switched-off module [11]. It is straightforward that the trade off between reliability, power and throughput depends on the final application requirements.
We therefore obtained the following objectives:

1. scalable architecture that can be easily adapted to the mission requirements
2. high throughput due to the highly parallel architecture
3. capability to reduce the power and failure rate of the memory by simply turning off the unused modules
4. graceful degradation of the system both in terms of functionalities performed by each memory module and of availability of the service
5. protection of the implementation of each function using different fault tolerant techniques.

For deciding the design strategies, we evaluated each function in terms of the impact of its failure on the overall performance of the SSMM system. In particular, we evaluated both the impact of permanent and transient faults after their detection.
The functionalities of a block affected by a transient fault can be recovered, after its detection, simply reinitializing the hardware block. Therefore the impact of a transient faults is mainly related to the stored data integrity.
On the other hand, permanent faults cause the unavailability of one of the implemented functions; therefore, their impact is mainly related to the performance assessment.

	Write	Read	Delete
Transient faults	Critical	-	Critical
Permanent faults	-	Critical	-
Latency	Critical	-	Critical

Table I: transient and permanent fault impact on file system functions

The results of these evaluations are reported in Table I.
The table can be explained as follows: the occurrence of transient faults on the functions that access the memory in write mode (like write and delete) are critical because they can cause unrecoverable incongruences on the FAT of the module and consequent data loss. Instead, transient failure occurrence on read function has no impact in terms of data integrity while causes a possible delay in the accomplishment of the function. In fact, after a temporary fault is detected on the read function, a message to the user signals the possible corruption of sent data. The re-initialization of the faulty hardware block allows reading the correct data. The impact of permanent faults was evaluated with respect to degradation of the system functionalities.
It is obvious that the loss of the read function has a greater impact on a module than the other functions, since it is useless writing data on a memory module that cannot be read.
Finally, to choose the fault tolerant technique to be applied to each file system basic function, some consideration about the fault detection latency must be drawn. In fact, some blocks must guarantee low detection latency in order to avoid unrecoverable effects, while other blocks have no strict latency requirement. For the latter blocks a detection based on signature analysis allows low area overhead, for the former ones the detection is based on Concurrent Error Detection (CED) techniques [14].
To tolerate permanent fault different choice can be made. Triple Modular Redundancy (TMR), allows both detection and correction with low latency and high area overhead.
The use of CED with a cold spare module allows better reliability (the failure rate of the cold spare is 1/10 of an active one) with a area overhead slightly lower than the TMR and a down-time needed to turn on the spare module. Signature analysis, with a cold spare module, allows low area overhead at the cost of both higher error detection latency and down-time.

4. FAULT TOLERANT FILE SYSTEM IMPLEMENTATION

In Table II the area occupancy of the write, read and delete hardware blocks in terms of FPGA Complex Logic Blocks (CLBs) are summarized. The above results have been obtained for a XCV1000 Xilinx Virtex™ device by using Synplify™ [13] as a synthesis tool. By using the required level of reliability, performance, power consumption and the results of Table I, for each basic block, *ad hoc* fault tolerant detection and redundancy techniques can be chosen.

	Write	Read	Delete
# CLB	652	260	178

Table II: area occupancy of FAT functions

Table III and Table IV show two possible sets of fault tolerant techniques that can be applied to the file system functions.
In Table III (Set A) CED techniques are applied to the write and delete functions reducing the latency of the transient and permanent fault detection that could lead to irreparable FAT incongruities according to the results reported in Table I. This set of solutions has a limited area overhead because no spares are introduced. The drawback of this choice is that when a permanent fault is detected, the function can't be recovered and the performance of the mass memory is degraded.

	Write	Read	Delete
Detection technique	CED	Signature Analysis	CED
Redundancy	None	None	None
# CLB	1015	390	265

Table III: characteristic of Set A

In Table IV (Set B) CED is applied to both write and delete functions while TMR is applied to read function to provide better reliability and lower read latency.

	Write	Read	Delete
Detection technique	CED	-	CED
Redundancy	Cold spare	TMR	Cold spare
# CLB	1630	803	445

Table IV: characteristic of Set B

Lower read latency is a requirement for Low Earth Orbit (LEO) satellite. In fact, the short window of visibility requires the implementation of fast downloads with no repetition. This set of solutions has a higher area overhead with respect to the previous one. On the other hand, when a permanent fault is detected, the functionality can be recovered using the redundant blocks and the performances of the mass memory are not degraded. Depending on the selected set of solutions different level of reliability can be obtained, as is described in the following section.

5. PERFORMANCE EVALUATION

The performance evaluation of the system can be done both in terms of reliability and throughput. The graceful degradation capability can be evaluated by calculating the probability that the system works correctly at different level of performance. This approach is quite similar to the performability defined in [12]. Given a certain failure rate λ of a single CLB we can estimate the reliability of the various file system functional blocks for the FPGA implementation. We assume that the reliability of each block is the reliability of the series of the CLB needed to implement the function. Therefore, the failure rate of the functions can be expressed as follows:

$$\lambda_W = 652*\lambda;\ \lambda_R = 260*\lambda;\ \lambda_D = 178*\lambda;$$

Where: λ_W, λ_R, λ_D are the failure rates of the write, read and delete blocks, respectively.

In a configuration with only fault detection capability (Set A), inside the memory interface, the reliability can be assumed as the reliability of the series of the function blocks. The failure rate λ_{MI} of a single memory interface is:

$$\lambda_{MI} = \lambda_W + \lambda_R + \lambda_D;$$

Using this failure rate we estimate the reliability of the overall set of memory interfaces at different level of performances for a 4 Gigabytes SSMM composed of four 1 Gigabytes memory modules. The possible levels of performance are defined as the amount of memory available. If we have n interfaces, the reliability with r interfaces functioning is:

$$R_s = \sum_{i=r}^{n} \binom{n}{i} [R(t)]^i [1 - R(t)]^{n-i}$$

Where $R(t)$ is the reliability of a single interface.

If an unrecoverable fault is detected, the SSMM is reconfigured reducing the available storing capability. In Fig.3 we draw the reliability curves. We can notice that the reliability to a typical End of Mission (EOM) of 36 months is about 70% if complete availability is required (4 GB), while, in case of 75%, 50 % and 25% storage capability the reliabilities are above 95%.

The use of redundancy inside the memory interface (Set B) provides better reliability for each memory interface.

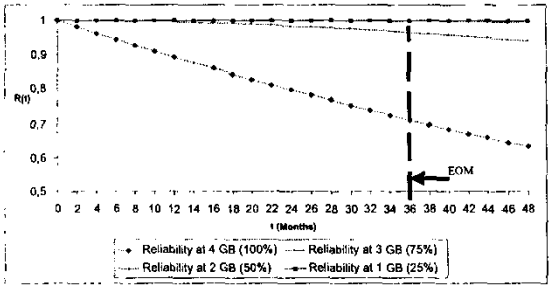

Fig.3 - Reliability curves for different performances

In fact, with this configuration and the same failure rates for each hardware block (λ_W λ_R λ_D) we can apply the formulas given in [14] to calculate the reliabilities of a TMR configuration and cold spare configuration.

Using the obtained reliabilities of the hardware blocks the reliability curves of Fig.3 can be recalculated. In Table V we show the improvement of the reliability for each performance configuration at the EOM.

R(t) t=3 years	4 GB	3 GB	2 GB	1 GB
Set A	0,7097	0,9638	0,9979	0,9999
Set B	0,9893	0,9999	0,9999	0,9999

Table V: reliability comparison @ EOM

6. CONCLUSIONS

In this paper we presented a hardware implemented file system manager for a fault tolerant SSMM.

We have described the methodologies used for the architectural design. In particular, we have evaluated the impact of faults and their detection latency for each block. Starting from these evaluations we applied the most suitable fault tolerant techniques. We evaluated the reliability of a configuration of 4 GBytes SSMM with respect to the improvement gained with the introduction of the selected techniques. Partitioning the system into four independent modules we obtained graceful degradation capability in terms of storage performances. Finally, to improve the overall reliability we introduced suitable redundancies on each hardware block.

7. REFERENCES

[1] M.P. Kluth, F. Simon, J.Y. Le Gall, E. Muller, "Design of a fault tolerant 100 Gbits solid-state mass memory for satellites", VLSI Test Symposium, 1996., Proceedings of 14th , 1996.

[2] Fichna, T.; Gartner, M.; Gliem, F.; Rombeck, F. "Fault-tolerance of spacebome semiconductor mass memories", Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on , 1998 .

[3] M. Ottavi, G.C. Cardarilli, P. Marinucci, S. Pontarelli, M. Re, A. Salsano, "Development of a dynamic routing system for a fault tolerant solid state mass memory", ISCAS 2001.

[4] SpaceWire Home Page http://www.estec.esa.nl/tech/spacewire/index.html

[5] D. Maeusli, F. Teston, P. Vuilleumier, R. Harboe-sorensen "ESA Developments In Solid Sate Mass Memories", in ESA publication

[6] S.M. Parkes, "Spacewire: The Standard", DASIA'99

[7] Mahmood, A.; McCluskey, E.J. "Concurrent error detection using watchdog processors-a survey" Computers, IEEE Transactions on , Volume: 37 Issue: 2 , Feb. 1988 pp 160 -174

[8] Saxena, N.R.; McCluskey, E.J. "Parallel signature analysis design with bounds on aliasing" Computers, IEEE Transactions on , Volume: 46 Issue: 4 , April 1997 pp. 425-438

[9] G.C. Cardarilli, P. Marinucci, A. Salsano, "Fault-tolerant Solid State Mass Memory for Satellite Applications", IMTC'98

[10] G.C. Cardarilli, P. Marinucci, S. Bertazzoni, M. Salmeri, A.Salsano, "Design of Fault-tolerant Solid State Mass Memory",DFT'99.

[11] Military- Handbook 338B

[12] J. F. Meyer, "On evaluating the performability of degradable computing systems," IEEE Trans. Computer, vol. C-29, pp. 720-731, Aug. 1980.

[13] Synplicity Home Page http://www.synplicity.com

[14] Parag K. Lala, "Fault tolerant and Fault Testable Hardware Design", Prantice Hall, 1985.