TOC-BISR: A SELF-REPAIR SCHEME FOR MEMORIES IN EMBEDDED SYSTEMS

Gustavo Neuberger, Fernanda Lima Kastensmidt, and Ricardo Reis
Universidade Federal do Rio Grande do Sul (UFRGS) - Instituto de Informática
CP15064, CEP91501-970, Porto Alegre, Brazil email: {neuberger,fglima,reis}@inf.ufrgs.br

Abstract: Memories are important components in embedded systems, since complex systems require more and more amount of data storage. Upcoming memories are more and more required to guarantee reliability for secure applications in the presence of massive soft and hard errors. This work proposes a fault tolerant customizable technique that combines EDAC, which can correct soft errors, and a built-in self-repair approach based on online testing and a Content Addressable Memory (CAM), which can tolerate hard errors. The goals of this approach are ensuring the correct operation of the system, extending the lifetime of the component, and improving the yield. This digital system was described in VHDL and synthesized in FPGA. The approach is customizable in terms of EDAC code, test algorithm and CAM size. The main advantage of the customization is to choose the best tradeoff between the number and type of tolerated and corrected errors compared to the area overhead and performance penalties for a target application.

Keywords: Self-Repair, Embedded Memories, Fault-Tolerance, EDAC Code

1. INTRODUCTION

The technological evolution to very deep submicron processes with drastic device shrinking, power supply reduction, and increasing operating speeds has significantly increased the sensitivity of integrated circuits to multiple faults[1]. Faults can be classified by their effects: transient or permanent. In this paper, transient faults will be identified as soft errors and permanent faults as hard errors. Soft errors occur due to radiation effects and noise. Technological scaling increases device vulnerability to radiation, as

charged particles that were once negligible are now much more likely to produce upsets at ground level[2]. Hard errors occur due to manufacturing defects and, more recently, radiation effects, such as total ionization dose or heavy ion strike, which can provoke faults with permanent effects[3, 4, 5].

One of the main advantages of developing fault tolerant techniques for soft and hard errors is to keep the memories operating in hostile environments without being replaced, even in the presence of multiple faults. Techniques usually are able to cope with one or other type of error and in many cases they are only able to detect them. The challenge is to develop a methodology that can be easily applied to commercial memories to detect and correct soft and hard errors, or at least cope with them.

Many fault tolerance techniques have been proposed to detect and correct soft errors, such as, error detection and correction codes (EDAC). Hamming code[6] and Reed-Solomon code[7] are good examples of high-level design techniques able to correct single and multiple soft errors. However they have some limitations. The first drawback of EDAC codes is their inefficiency in protecting rarely executed code addresses or data, for instance, error handling routines. In this case, latent errors remain undetected during a long period of time and consequently upsets can accumulate. This can overcome the protection of many fault-tolerant techniques that are based on single error detection and correction. An upset in this case must be detected and corrected before a second one occurs. Even the fault-tolerant architectures based on multiple upset corrections must periodically remove the latent errors. It must avoid a state where there are so many upsets that can no longer be corrected. The second deficiency of EDAC codes is that, although they are used to detect and tolerate hard errors in commercial memories, they can not eliminate the cause of the hard error, and in presence of both types of errors, EDAC codes will fail in the case of a soft error in the same address as a hard error.

Hard errors can be detected by a test that alters the memory contents continually. For detecting defects during the memory lifetime, transparent online testing techniques[8, 9] have been used. However, these techniques are not able to tolerate the detected defects. This drawback is usually ignored because once the memory is tested, all defects due to fabrication process are detected, and only soft errors will affect the memory. However, in very deep submicron technologies, hard errors can often appear during the lifetime due to radiation, which makes it necessary to develop a fault-tolerant technique able to tolerate hard errors. In addition, memories able to tolerate a set of defects can help to reduce yield losses, in such way that memories with pre-mapped defects can be sold in the market, as the use of fault tolerance techniques will be able to cope with these hard errors. This can reduce the cost of manufacturability.

Current methods found in literature are not able to cope with soft and hard errors at the same time during the normal memory operation, which is crucial today for high-reliability memories. This work proposes a fault-tolerant customizable technique that combines EDAC, which can correct soft errors, and a built-in self-repair approach based on online testing and a Content Addressable Memory (CAM), which can tolerate hard errors. The objectives of this methodology are to guarantee the correct operation of the device in the presence of soft and hard errors, extending the lifetime of the component by tolerating hard errors due to radiation, as well as improving the yield, by tolerating manufacturing defects.

This digital system was described in VHDL and prototyped in a Xilinx FPGA. The approach is customizable in terms of EDAC code, test algorithm and CAM size. The main advantage of having a customizable fault-tolerant methodology is to choose the best tradeoff between the number and type of tolerated and corrected errors compared to the area overhead and performance penalties for a target application.

This paper is organized as follows: section 2 shows the related work on tolerant techniques against soft and hard errors. Section 3 describes the developed built-in self-repair system, the detailed implementation of the method in an embedded system programmable platform and the advantages of the customization of this platform in terms of flexibility. The area and performance results are evaluated in section 4. Conclusions and future work are discussed in section 5.

2. MOTIVATION

With the development of the IC technology, complex systems can be built in a single chip (System-on Chip, SoC). Modern FPGAs include microprocessors, memories and several other specialized components, and a full system can be designed in a single FPGA. Even a very large system that uses external microprocessors and memories can use an FPGA to implement a set of more complex specialized functions. The main benefit of an FPGA is the re-programmability, useful in several applications. For example, in space applications, reprogramming the FPGA after the launch, to correct errors, improve performance or increase features, can reduce the mission cost. However, in general, the systems need big amounts of memory. In space applications, and also at ground level, the memories are the component most susceptible to faults.

Considering the case of a system composed by a microprocessor, a memory and several hardware cores running in parallel in an FPGA, the memory can have its reliability increased with only the inclusion of one

more core in the FPGA, responsible to test and repair the memory. This solution can be very attractive, since it does not increase the cost, if the FPGA still has available enough logic blocks to implement the test and repair system, impacting only the performance and power. Figure 1 shows the scheme of this possible system.

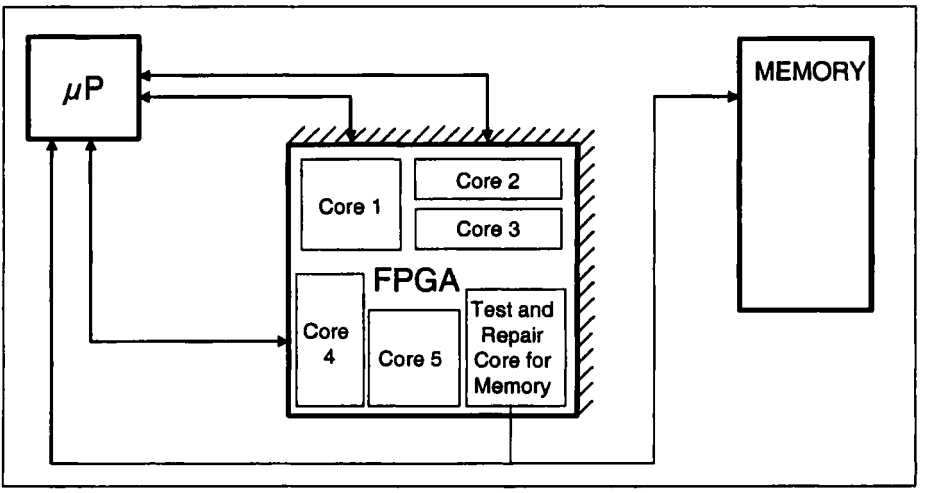

Figure 1. Scheme of a System Composed of Microprocessor, Memory and FPGA

3. RELATED WORK

Recent works have been concerned about the effects of multiple faults in memories[10]. Multiple soft errors can occur in VDSM memories due to the incident angle of the charged particle, to the logic cell density or to the time between faults. If the time between faults is too short, faults can accumulate in the memory. Consequently, standard error correction codes such as Hamming code may not be enough in many applications.

Thaller[11] has proposed the Transparent Online Memory Test (TOMT) for soft and hard errors detection. It combines parity or Hamming code with March tests to perform active fault detection over the whole memory, warning the system. A March test consists of a sequence of March elements. A March element consists of a sequence of operations applied to every cell, in either one of two address orders: increasing (⇑) or decreasing (⇓).

The TOMT is completely transparent for the user. It does not affect the memory usage by the processor. The drawback of this scheme is that it only detects the faults, sending an error message to the processor when a problem occurs. The scheme is not able to correct on-the-fly the hard error and to

continue the user's application. Figure 2 shows the schematic of the TOMT unit used to perform the active test in the memory. The system is placed between the processor and the memory and it is transparent to both.

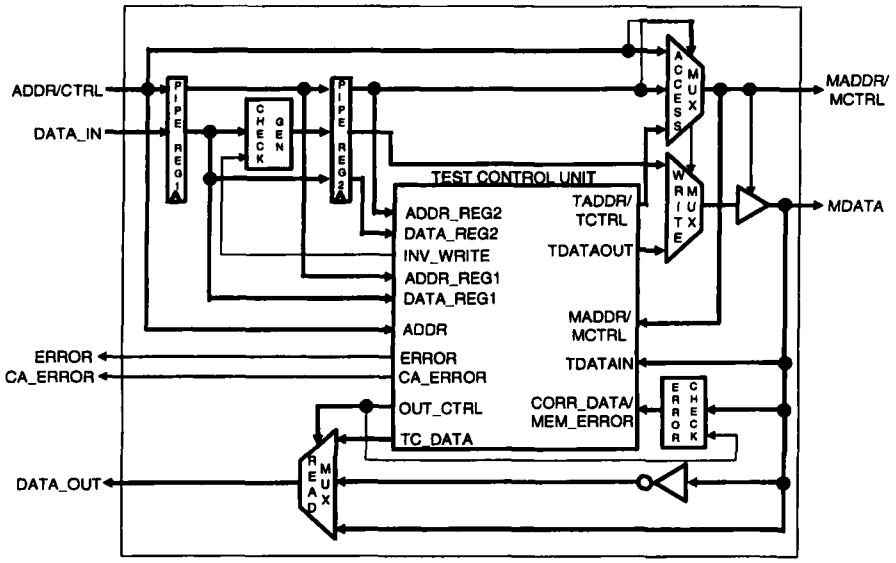

Figure 2. TOMT Unit Schematic[11]

A BISR system is proposed by Zappa[12], using a programmable BIST together with a Flash memory to store the defective words. The BIST can be programmed with different March tests, to detect different types of hard faults. However, as this system does not perform online BISR, it must stop the operation of the SRAM to make the test. Another deficiency is its incapacity to handle soft errors. Figure 3 shows the architecture of this system.

As seen previously, recent published related works are not able to detect and correct soft errors at the same time they are able to cope with hard errors. Next section presents an online built-in self-repair (BISR) customizable for memories able to detect and correct soft errors and to tolerate hard errors in VDSM memories, ensuring the correct operation of the application without interruption.

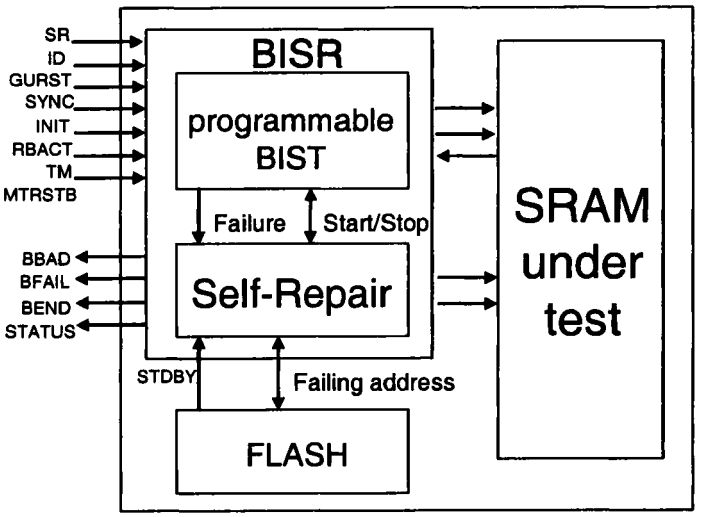

Figure 3. BISR Scheme using a Flash Memory

4. DEVELOPING A CUSTOMIZABLE ONLINE BUILT-IN SELF-REPAIR SYSTEM

The proposed Transparent Online Customizable Built-in Self-Repair system, named TOC-BISR, is based on the TOMT unit[11], with a set of modifications in order to tolerate the soft and hard errors instead of only warning the processor in the case of occurrence of them. The new BISR method is composed of:

- Test control unit presented by Thaller[11], that controls the March test and the generation of addresses to send to the memory; which was re-implemented in a programmable platform for further customization,
- EDAC encoder and decoder blocks, to correct soft errors in the read data;
- Content-Addressable Memory (CAM), to store redundant addresses to tolerate hard errors;
- Extra logic (inverters and multiplexers) to route the data from/to the processor and the memory.

The test control unit uses the same algorithm as TOMT[11], being composed of a word-level algorithm and a bit-level algorithm, performing a March test in the whole memory. It is important to emphasize that the processor always has the priority to access the memory over the test algorithm. However, if the processor wants to read the same address that is currently under testing, it must read from the backup register instead of the

memory. At this moment, the data stored in the memory can be wrong, because it can be in the middle of the March algorithm.

In order to cope with soft errors, an EDAC encoder and decoder must be placed in series with the data path, differently than in TOMT[11], where parity or Hamming is used in parallel only to detect the error. Any EDAC code can be used, but in the first implementation Hamming code was used to simplify the encoding and decoding circuits. In the future, it is planned to use other codes to increase the fault tolerance to multiple errors, like Reed-Solomon codes.

Concerning tolerance to hard errors, a possibility is to remap the faulty address into another address, or extra memory area. In this work, a CAM is used to store the data together with the faulty addresses. When the processor requires a certain address, both the main memory and the CAM are read in parallel. If the address is found in the CAM, the data stored in the CAM is sent to the processor; otherwise, the data stored in the memory is used. The figure 4 shows what is stored in the CAM, and CAM inputs and outputs.

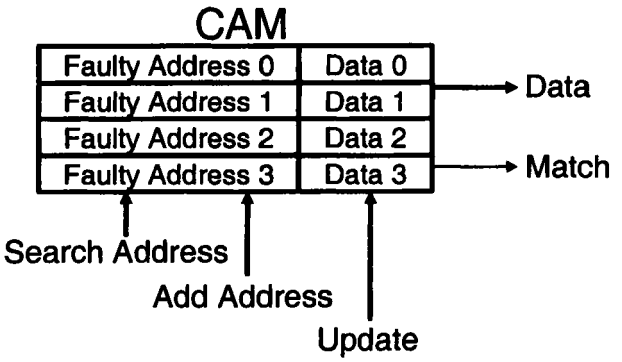

Figure 4. Content-Addressable Memory Scheme

The search address signal, figure 4, is used to check if an address is present in the CAM. If yes, then the match signal goes high, and the data signal contains the data associated to that address. This means that the current address requested by the processor correspond to a location with a hard error in the main memory, which has been previously detected by the test algorithm (in this case, March). The data stored in the memory is ignored, and the data stored in the CAM is used instead. The *add* address signal inserts a new faulty address in the CAM. The insertion occurs when the March test finds a hard error in the main memory. Finally, the update signal is used to write a new data in an address that is faulty. In this case, the data is updated in the current address stored in the CAM.

Figure 5 shows the final schematic of the TOC-BISR system. In addition to all blocks that were inserted in the original method presented in TOMT[11]: the CAM, Hamming encoder and decoder, and CAM mux; the error signal of the test control unit now must be sent to the CAM instead of being an error signal to the processor, as well as the faulty address. The final proposed method is capable of tolerating soft and hard upsets.

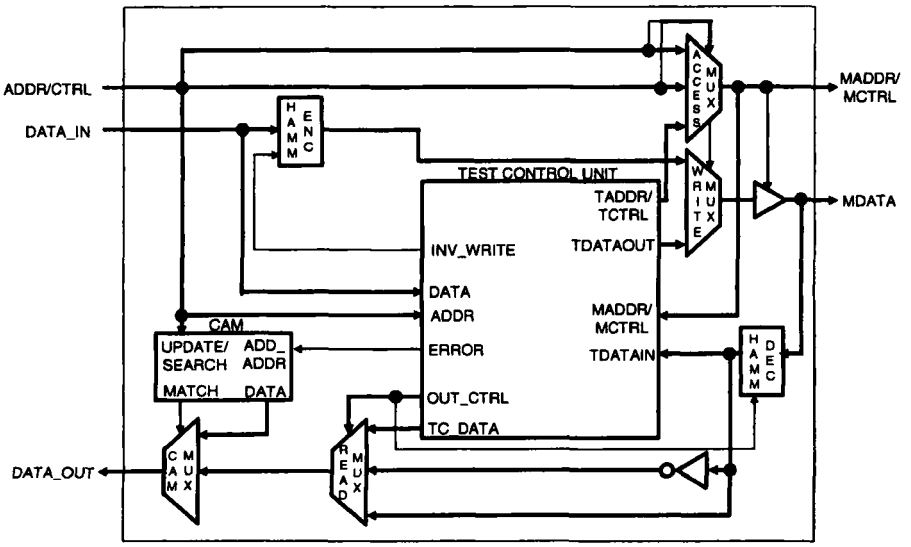

Figure 5. TOC-BISR System

The use of a configurable platform offers additional advantages for the customization of the TOC-BISR. The reconfigurability of FPGAs can be used to increase the features of the TOC-BISR unit, such as: changing the EDAC code to increase the tolerance to multiple soft errors, the March algorithm to increase the fault coverage of hard errors, and the size of the CAM to increase the tolerance to a maximum number of addresses containing hard errors. It would be possible to use the whole available set of logic of the FPGA to implement the TOC-BISR unit, with the largest possible CAM, a complex EDAC code and the most efficient March test. However, it is much more attractive to have a programmable platform to attend the needs of each case, or extra power could be dissipated and performance lost without reason. For example, the system could be initialized with a very small CAM, and only if the CAM becomes full, it is reprogrammed to have a larger size.

The TOC-BISR unit implemented in an FPGA can be integrated transparently in a system, being placed between the processor and the main memory. The figure 6 shows the TOC-BISR unit integrated in a system.

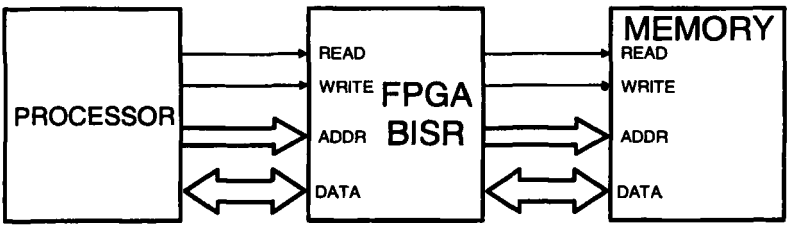

Figure 6. System Integration of the TOC-BISR

For evaluation purposes, the TOC-BISR customizable method was described in VHDL and implemented in a Xilinx FPGA (VirtexE family). An 8-bit address was used, with 16-bit data, and 16 available positions in the CAM. There is no pre-defined CAM component in the Virtex FPGA. There are solutions to implement a CAM using embedded RAM memories (BlockRAMs) available in most FPGA families from Xilinx[13].

All the TOC-BISR circuit must also be protected against soft and hard errors by definition as well. Normally in an ASIC, the sensitive parts to soft errors are mainly the memory elements: the registers inside the test control unit and the CAM. But all the logic can be sensitive to hard errors. Consequently, the most appropriate solution for protection is the Triple Modular Redundancy (TMR). In the case of the FPGA, TMR is also the most suitable method to protect against soft and hard errors. According to the circuit, there are also other methods that can be used for fault tolerance in the FPGA and they can be applied in the TOC-BISR[14].

5. EXPERIMENTAL RESULTS

Some comparisons were made with different implementations using the FPGA platform, after the design of the TOC-BISR unit and its description in VHDL. The comparisons are in terms of online testing and soft error tolerance approaches, aiming at evaluating the drawback of fault tolerance efficiency compared to area and performance overhead. The results shown in this section are based in the synthesis of the TOC-BISR VHDL code in the Xilinx FPGA VirtexE-XCV300EPQ240[15]. Although the TOC-BISR core was developed to protect any commercial memory, in the experimental result, the protected memory is embedded in the same FPGA that contains the TOC-BISR approach.

Table 1 shows the results of the clock period obtained in the usage of the memory by itself and with Hamming code. It also shows the area to implement the Hamming code. The area is given by number of Lookup Tables (LUTs). The results are for a memory with 8-bit address and 16-bit

data. In addition to the area for the encoder and decoder, the error correction code uses some extra RAM area for the parity bits. A 5-bit parity for a 16-bit word data is used in the Hamming code version.

Table 1. Results for standard and error correction code memories

	No Protection	Hamming EDAC
Clock Period	7.5 ns	13.2 ns
#LUT4	N/A	51

Table 2 presents the results of the TOMT unit[11] with Hamming code implemented in parallel for error detection only; a modified TOMT unit with Hamming code implemented in series, for autonomous soft error correction, and the proposed TOC-BISR unit. The table 2 shows the results obtained for these units.

Table 2. Results for TOMT and BISR Units

	TOMT[11]	TOMT[11] + Hamming	TOC-BISR
Clock Period	13.3 ns	17.63 ns	17.73 ns
#LUT4	463	443	680

The TOC-BISR unit does not change drastically the clock period compared with the previous unit, because the CAM and the main memory with Hamming protection are read in parallel, and the CAM access time is smaller than the Hamming decoding, not altering the total clock period. However, there is an increase in area of about 50% to implement the extra logic in the CAM, together with one BlockRAM, as already explained in previous section. In an ASIC implementation, the CAM can be more efficiently implemented in terms of area, with few SRAM cells and no extra surround logic, but it does not give the flexibility of size customization as the one proposed here.

The performance penalties seen are more than twice the period of the no-protected memory. This can be explained by the EDAC code insertion in the critical path, which happens with all memories protected by this technique. For large memories, the delay for the EDAC code can represent a small increase of the memory time access time in percentage. However, an increase of at least 50% in delay was already expected, and it can be a reasonable price to pay to ensure fault tolerance for soft and hard errors. In terms of area, the results are quite good, since it uses about only 10% of the available logic in the FPGA in the case of VirtexE-XCV300.

The customization of the TOC-BISR can be done with little extra cost. For example, changing the EDAC to Reed-Solomon code, it would be expected to increase the delay and the area due to the code about 50% compared to the memory protected with Hamming code[11]. The March algorithm could be changed with no cost for the application. The algorithm

only changes the total time expended performing the test in the whole memory. Finally, the CAM can have its size increased with almost no cost in performance, and a linear increase in area cost.

6. CONCLUSIONS

This paper has presented a transparent online customizable built-in self-repair (TOC-BISR) system to protect commercial memories against soft and hard errors. The first implemented version uses Hamming code combined with a March test algorithm together with a CAM, which is used to remap addresses with permanent faults.

Results show that the presence of the CAM did not significantly increase the clock period compared to a similar approach that performs only soft error correction and online testing. The main advantage of the proposed approach is the flexible customization after the design. New EDAC codes and testing algorithms can be used to update and improve the fault tolerance of the system, as well as, the CAM size can be updated to tolerate a large number of hard errors in the memory. Future works include the study of some parallelism techniques in order to reduce the performance overhead, and the investigation of other options to store the defective addresses, such as using a Flash memory.

REFERENCES

1. K. Johansson, B. Dyreklev, B. Granbom, M. C. Calvet, S. Fourtine, O. Feuillatre, "In-Flight and Ground Testing of Single Event Upset Sensitivity in Static RAMs", IEEE Transactions on Nuclear Science, Vol. 45, pp. 1628-1632, 1998.
2. A. Johnston, "Scaling and Technology Issues for Soft Error Rates", 4th Annual Research Conference on Reliability, October, 2000.
3. T. R. Oldham, K. W. Bennett, J. Beaucour, T. Carriere, C. Polvey, P. Garnier, "Total Dose Failures in Advanced Electronics from Single Ions", IEEE Transactions on Nuclear Science, Vol. 40, pp. 1820-1830, 1993.
4. P. Cheynet, R. Velazco, R. Ecoffet, S. Duzellier, J. P. David, J. Loquet, "Comparison Between Ground Tests and Flight Data for Two Static 32KB Memories", Proceedings of European Conf. Radiation and Its Effects on Components and Systems, pp. 554-557, 1999.
5. J. P. David, J. Loquet, S. Duzellier, "Heavy Ions Induced Latent Stuck Bits Revealed by Total Dose Irradiation in 4T Cells SRAMs", Proceedings of European Conf. Radiation and Its Effects on Components and Systems, pp. 80-86, 1999.
6. R. W. Hamming, "Error Detecting Codes", Bell System Technical Journal, Vol.26, no. 2, April 1950.
7. A. D. Houghton, "The Engineer's Error Coding Handbook", London: Chapman & Hall, 1997.

8. D. C. Huang, W. B. Jone, S. R. Das, "An efficient parallel transparent BIST method for multiple embedded memory buffers", Proceedings of International Conf. VLSI Design, pp. 379-384, 2001.
9. M. G. Karpovsky, V. N. Yarmolik, "Transparent memory BIST", Proceedings of International Workshop on Memory Technology, Design and Testing, pp. 106-111, 1994.
10. R. A. Reed, M. A. Carts, P. W. Marshall, O. Musseau, P. J. McNulty, D. R. Roth, S. Buchner, J. Melinger, T. Corbiere, "Heavy Ion and Proton Induced Single Event Multiple Upsets", IEEE Transactions on Nuclear Science, Vol. 44, Issue 6, pp. 2224-2229, December 1997.
11. K. Thaller, A. Steininger, "A Transparent Online Memory Test for Simultaneous Detection of Functional Faults and Soft Errors in Memories", IEEE Transactions on Reliability, Vol. 52, Issue 4, pp. 413-422, December 2003.
12. R. Zappa, C. Selva, D. Rimondi, C. Torelli, M. Crestan, G. Mastrodomenico, L. Albani, "Micro Programmable Built-In Self Repair for SRAMs", Proceedings of International Workshop on Memory Technology, Design and Testing (MTDT), pp. 72-77, 2004.
13. H. T. Vergos, D. Nikolos, "Performance Recovery in Direct-Mapped Faulty Caches via the Use of a Very Small Fully Associative Spare Cache", Proceedings of International Computer Performance and Dependability Symposium, pp. 326-332, 1995.
14. F. L. Kastensmidt, G. Neuberger, L. Carro, R. Reis, "Designing and Testing Fault-Tolerant Techniques for SRAM-based FPGAs", ACM Computer Frontiers Conference, 2004.
15. Xilinx Inc., "Virtex™ 2.5 V Field Programmable Gate Arrays", Xilinx Datasheet DS008, v2.4, October, 2000.