

Available online at www.sciencedirect.com**ScienceDirect**

Procedia Computer Science 79 (2016) 827 – 834

Procedia
Computer Science

7th International Conference on Communication, Computing and Virtualization 2016

FPGA Based Multiple Fault Tolerant and Recoverable Technique Using Triple Modular Redundancy (FRTMR)

Shubham C. Anjankar^{a*}, Dr. Mahesh T. Kolte^b, Ajinkya Pund^c, Pratiksha Kolte^d, Ankita Kumar^e, Pranav Mankar^f, Kunal Ambhore^g

^aAssistant Professor, Ramdeobaba College of Engineering, Nagpur 440013, MH, INDIA

^bProfessor, MIT College of Engineering, Pune 411038, MH, INDIA

^cAssistant Professor, G.H. Rasoni College of Engineering, Nagpur, MH INDIA

^dPVGOET, Pune, MH INDIA

^eAssistant Professor, Ramdeobaba College of Engineering, Nagpur 440013, MH, INDIA

^fAssistant Professor, G.H. Rasoni College of Engineering, Nagpur, MH INDIA

^gMIT College of Engineering, Pune 411038, MH, INDIA

Abstract

Triple Modular Redundancy (TMR) is real time reliability known to improve computing systems. This paper presents an approach towards the implementation of a fault tolerant FPGA based technique. Proposed scheme allows the diagnosis of transient and permanent fault affecting systems. This technique allows to easily identify whether a fault affects one of the replicated modules or the technique itself and whether such a fault is permanent or transient. The scheme can effectively recover computing systems from single transient fault w/o introducing any re-computing delay; hence, it is suitable for real time applications. Time taken for identification of the fault and recovery is less than 8ns. FPGA based fault injection experiment reveal that, the error detection and recovery coverage of 100% and 90% in the presence of single and two faulty models, respectively.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Peer-review under responsibility of the Organizing Committee of ICCCV 2016

Keywords: Fault Tolerance; TMR ; Reliability ; real-time; re-computing; delay;

1. Introduction

At present, the use of embedded systems in safety - critical application such as, aviance, process control and patient

* Corresponding author. Tel.: +91-992-370-0538.

E-mail address: anjankarsc1@rknc.edu

life supporting monitoring has become a common trend. Fault and error safety critical systems has a timing constraint and fault tolerant requirement ^[1, 2]. Electronic systems, especially those based on FPGA (Field Programmable Gate Array), are prostrate to fault due to many factors, like operations in harsh environment as encountered in space or nuclear applications ^[5]. Increasing complexity of these applications puts up ever stronger demands on the performance and reliability of the used system. To meet the reliability requirement, such embedded system should be equipped with appropriate error detection and meeting the timing constraint limitation are (negative coefficient) conflicting objectives. The reliability improvement may have negative impact on timing constraints. For Example Rollback Recovery ^[15, 16].

For safety critical application, without considering its real constraint, enhancing the reliability is not justifiable. Hence, for stipulating, Fault tolerant techniques with minimum performance overhead is embedded processor is of decisive importance ^[1].

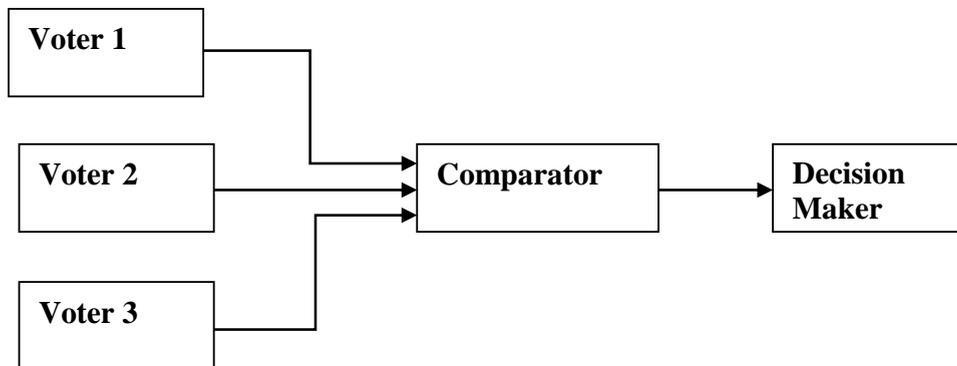


Fig. 1. TMR Block System

Modular Redundancy is the most popular approach to achieve the reliability and the timing constraints ^[4]. Modular Redundancy employs multiples replicas of the same module. Error occurring at the replicated modules. One of the approach to provide fault tolerance in real time systems is through redundancy, such as N- modular redundancy (NMR) or duplex pair. An NMR system replicates a computing source into parallel running N- module and uses voter to mask errors at output. Hence, for NMR it is possible to successfully tolerate the fault as long as it happens in no more than $(N/2)$ modules. In duplex pair system it uses two pair of duplicated modules (duplexes), which are four in total ^[6, 3]. The duplication of modules is for error detection purpose and duplex as used in pair so that when an error is detected in one of the duplex module the output from the other module will be selected as output of the system could still cont. producing current data. Though NMR and Duplex pairs can enhance reliability of system, only effective for short mission time unless repair is possible.

One of the widely used and commonly known used fault tolerant modular redundancy technique is triple modular redundancy. Traditional TMR system consisting of three redundant module voter at the modules output has some short coming which should be addressed in order to be implied in safety critical application .One of the major (disadvantage) short coming of the traditional TMR is its inability to cope with TMR failures. TMR failure is with the reference to the failure in TMR system causes by multiple fault in modules or for faulty voter. In roll back recovery system operation is block up to some points in its performance when an error occur and is detected during operation, the system will restore in its back to the previous check point and recomputed the system using TMR for recovery ^[16, 17]. However time overhead is introduce due to re-computation. Another way to obtain correct states is real-forward recovery, which is used in duplex system. It copies correct states from the fault free redundant modules or a spare to the faulty modules to avoid re-computation. However duplex system with spare are unable to provide sufficient redundancy. A quadplex (duplex pair) system provide enough redundancy so that no single fault will receive re-computation. However, draw in quad duplex system is its high cost. As while state in the faulty module are being restored, during that time other fault free modules are still operational ^[6, 9]. IT is suitable for and used in quradplex system but may not be suitable and used in TMR system. Reason behind it is that quradplex system will work properly while one is of restored but the TMR with reaming two module will not be able to provide error masking.

In this paper, we present new approach to provide fault tolerance and recovery system which is known as Fault recoverable Triple Modular Redundancy (FRTMR). This recovery scheme uses roll forward approach to avoid re-computation and to meet strict transaction deadlines which addressed the short coming of traditional TMR [6]. The proposal technique called FRTMR has the ability to locate and remove latent fault using TMR modules as well as to recover the system from multiple fault. The concept and idea behind FRTMR is to reuse the available scan chains devoted specially for testability purpose. In order to compare the internal state of TMR modules to locate and restore the faulty states of faulty modules [1]. It is very suitable for real time application by synchronizing the three modules during recovery error masking is always preserved during computation.

The FRTMR technique has been analytically and experimentally evaluated and compared with the state of the art technique. As a core study the proposed technique has been implemented on cyclone III FPGA. The proposed analytical study shows that in the presence of multiple errors, FRTMR improves the reliability of TMR system. In addition to that the result are taken by injecting the fault in FPGA, The result of FPGA fault injection experiments demonstrate that FRTMR can detect and correct 100% and 90% of multiple fault effecting signal or double module [18,19,20].

2. Related Work

Consider sequential consisting of combinational and storage elements for computing module. As computing process in the computing module is real time task, it has to be finished with a deadline. In traditional TMR, only one faulty module is masked. Additional to that, the faulty module cannot be recovered in the traditional TMR as it cannot identified the faulty module. The technique prepared in the modified version of the voter to diagnose the faulty module [1, 5-8, 13]. Many prepared techniques uses hardware boned voter system, while some technique uses software boned method for voting and fault diagnose which some in the improper working, as the number of transient fault stores in the memory exceed the predefine values it treats that transient fault as the permanent fault [16, 18].

The use of high-performance COTS chips in spaceships, on the other hand, increases concerns about system reliability. This is due to high probabilities of transient faults as well as intermittent and permanent faults in these chips. A transient fault is caused by ionizing radiation, particle strike, or other external interference. The induced error can propagate and corrupt a software state but does not cause permanent damage to the hardware. In an experiment [9], injecting 50 MeV protons to three COTS devices caused data loss and device crash, but no symptom was observed that implied permanent damage of the devices (e.g., a high current condition). This type of faults is usually characterized as lasting up to one clock cycle. With the scaling of manufacturing technology, transients last longer because there is less space for the particle strike energy to dissipate. In current technology, transients can last up to two clock cycles or even longer [15, 8]. If scaling continues, there is a high chance this duration will become longer. An intermittent fault has a duration that ranges between a few to billions of clock cycles [11, 9]. This fault is caused by various physical characteristics of chips (e.g., irregularities in chip voltage and temperature). Transistors become more and more susceptible to such physical characteristics with the scaling of manufacturing technology and the integration of many transistors into a chip (e.g., for multi-cores and large caches). A permanent fault occurs as a result of a manufacturing defect or excessive use of transistors. Such excessive use degrades the reliability of transistor materials. This fault become like a long duration intermittent fault, but it permanently compromises the functionality of the transistors. Both intermittent and permanent faults are more common in high-density chips because of their high utilization and low operating voltage [10].

One of the effective ways of synthesizing a reliable computer system from unreliable COTS components is Triple modular redundancy (TMR). TMR can be implemented in software without a major modification in the COTS components. For example, N identical copies of the same program can be run, with each copy executing on a separate computing module. Here, the time between two consecutive voting operations is generally longer than it is in hardware-implemented TMR. In order to reduce the voting interval time, software TMR can not only use the final program output data but also the intermediate program states as voting data. Intermediate states include large size data (e.g., vector or matrix of floating-point data). This makes hardware TMR voter unsuitable for software TMR systems.

3. TMR Assembly

The TMR Assembly is composed of three processors (here dummy) with voters on all outputs. A voter is constructed by some simple logic gates and is able to find an error when inputs are not consistent. TMR design does

not have to deal with too many errors per unit time. The assumption of the TMR design is that we will not see identical errors on two processors at the same time. The voters pass the majority vote so, if the errors are identical, they will not be detected (and will, in fact, be turned into truth.)

3.1 One and Multi Bit Voter

The CFTP is designed to be fault tolerant by software. Its circuit needs to be able to detect an error by itself. In order to achieve that, the concept of a voter is generated. The function of a 1-bit voter has been introduced in Lashomb’s thesis. This section reviews the basic concepts and then starts constructing the TMR Assembly. Figure 2 shows what a 1-bit voter looks like. It is a simple circuit consisting of only AND and OR gates [7].

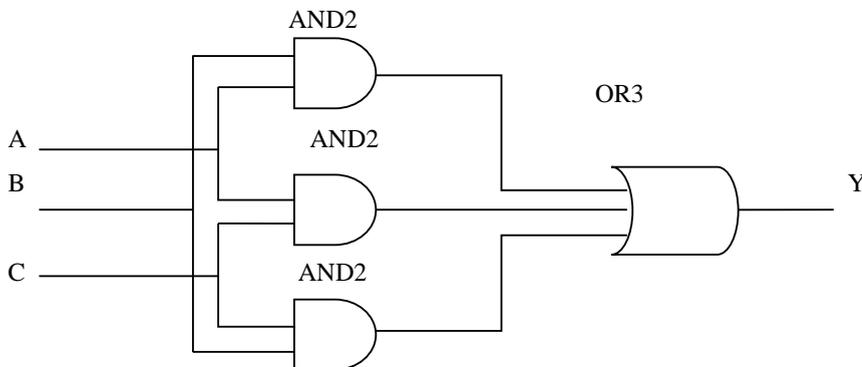


Fig. 2. 1-Bit Voter

The voter function is more obvious in the truth table shown in Table 1(a). This voter always selects the majority of identical bits as its output bit. If two or more inputs are incorrect, the voter output will also be incorrect. The ability to detect and correct two or more errors in a voter is not vital for a system (e.g., the CFTP).

Table 1. (a) 1-Bit Voter Output (b) 1 Bit Voter with ERR Output.

a

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b

A	B	C	Y	ERR
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Assuming a single error, the output is always correct, but we cannot tell if there has been an error just by looking at this output. Therefore, some extra gates are added to report the occurrence of an error. Figure 2 shows a voter with error detection and Table 1(b) is its truth table. The error detection, *ERR*, is 1 when one of the inputs is not identical

with the rest. When the CFTP is in space, it is possible to have an SEU on the voter itself. A bit flip may cause the voter output to be incorrect. Say the second column of Table 1(b) has a bit flipping on A. This flipping makes 1 become the majority bit and output Y will give a 1 not a 0. Since a voter is used to catch and correct an error, it is not pleasant if it has an error itself. Thus, some reliability is needed for the voter [10, 12, 14].

Table 2. Majority Voter Output

A	B	C	V_ERR	Y	D_ERR	CID_0	CID_1
0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1
0	1	0	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	0	1	0	1	1
1	1	1	0	1	0	0	0

3.2 Design Flow:

- Requirement Specification
- Functional Specification
- Block Diagram Designing
- Algorithm formulization
- Implementation of code in Verilog
- Simulation of the program

4. Simulations And Results

4.1 Compilation

All components generated for TMR design were simulated. Compilation of code generated for function is successful. Total logical element used is 25. Compilation is done for error correction as well as error detection code and it is successful

4.1.1 Analysis and Synthesis

Elapsed time required for Analysis and Synthesis is 0.5s and average processor used is 1, it uses 197MB virtual memory. Filter takes 0.17s and uses 1 processor, uses 272MB of virtual memory. Assembler takes 0.8s and uses 1 processor utilizing 211MB of virtual memory. All the details are shown in following Figure.

4.1.2 Simulation and Results

For one error in the input the simulation is done. The Change in Difference (CID) i.e. CID_0 is 0 and CID_1 is 1. Which show that the error is in input A. The 3rd bit of CID_1 is 1 which shows that the 3rd bit of input A is Faulty.

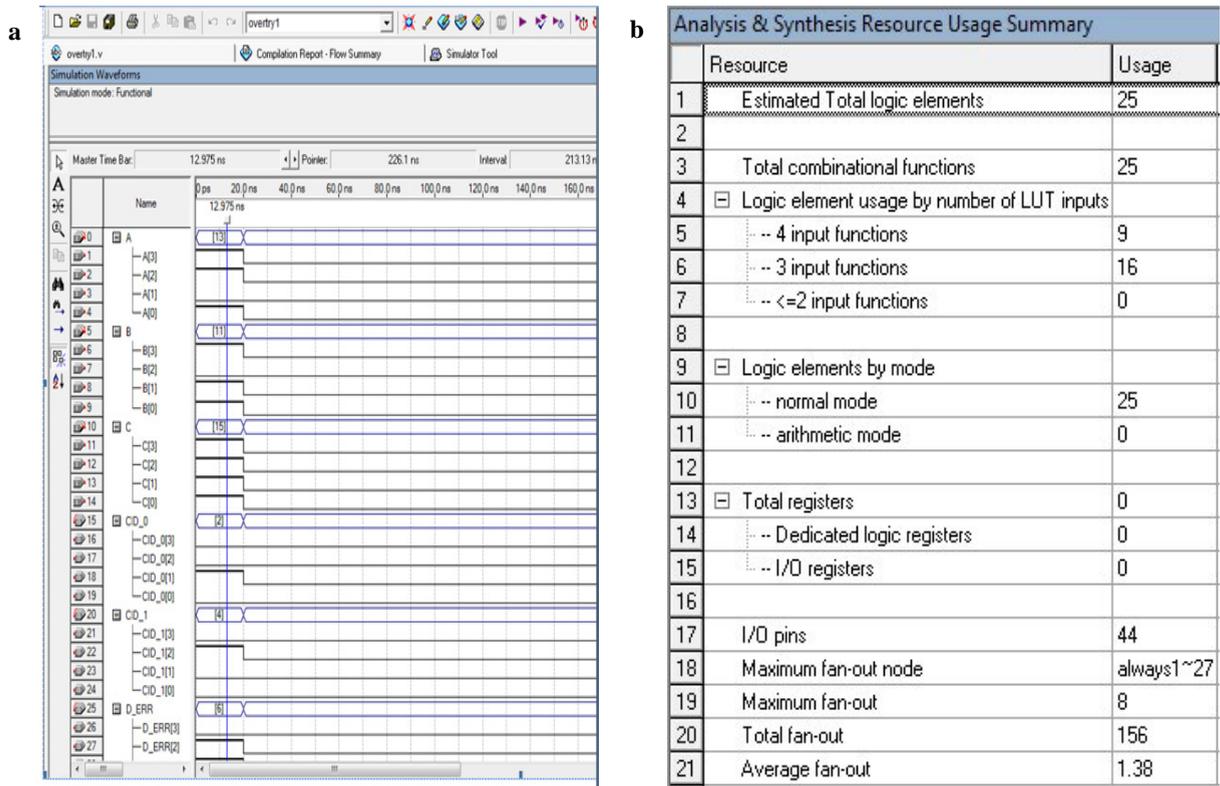


Fig. 3. (a) Output for two errors (b) Analysis and Synthesis

The Fault correction system detects that fault and assigns the corrected bit to faulty bit and corrected output is provided for working.

For two error in the input the simulation is done. The Change in Difference (CID) i.e. CID_0 is 0 and CID_1 is 1. Which show that the error is in input A. The 2nd and 4th bit of CID_1 is 1 which shows that the 2nd and 4th bit of input A is Faulty. The Fault correction system detects that fault and assigns the corrected bit to faulty bit and corrected output is provided for working.

Table 4. Flow Elapsed Time

Flow Elapsed Time					
	Module Name	Elapsed Time	Average Processors Used	Peak Virtual Memory	Total CPU Time (on all processors)
1	Analysis & Synthesis	00:00:05	1.0	197 MB	00:00:03
2	Fitter	00:00:17	1.0	272 MB	00:00:11
3	Assembler	00:00:08	1.0	211 MB	00:00:06
4	TimeQuest Timing Analyzer	00:00:06	1.0	223 MB	00:00:05
5	Total	00:00:36	--	--	00:00:25

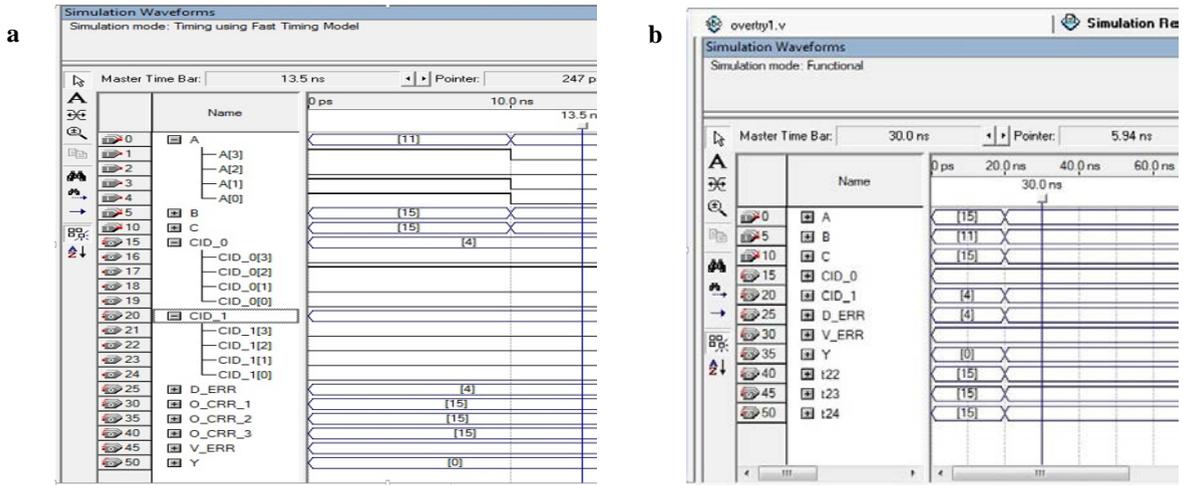


Fig. 4. (a) Output for Single error (b) Output for Single error with error corrector

For two errors in different input the simulation is done. The Change in Difference (CID) i.e. CID_0 is 0 and CID_1 is 1. Which show that the error is in input B. And CID-0 is 1 and CID_1 is 0 which shows that error is in input A. The 3rd bit of CID_0 is 1 which shows that the 3rd bit of input A is Faulty. The 3rd bit of CID_1 is 1 which shows that the 3rd bit of input B is Faulty. The Fault correction system detects that fault and assigns the corrected bit to faulty bit and corrected output is provided for working.

4.3. Time Require in Detection and Correction Using TMR

Time requires for correction and detection is shown in table. As compared to Cyclone II FPGA kit the time requires for execution the program is half on Cyclone III FPGA kit. The program is tested for different combinations of the inputs and errors. The result obtained by these testing are shown below. **All time is in ‘ns’ (Nano second)**

Table 3. Time Require in Detection and Correction Using TMR

	Cyclone III (DE0)		Cyclone II (DE2-70)	
	1	2	1	2
NO. of Fault/s	1	2	1	2
		Same Different		Same Different
Detection Delay Time	4.56	4.67 4.73	10.44	10.38 12.83
Corrected Output Delay Time	5.18	5.08 5.15	10.79	10.73 10.91

5. Conclusion

A failure in mission-critical system is either catastrophic or expensive to fix. In order to analyse the reliability of such systems, we have built a co-designed fault injector that can accurately emulate variable-duration faults in hardware. Our fault injection experiment using this tool showed two types of reliability problems in software-based

voters (i.e., used in N-modular redundancy). One software technique was designed to detect and tolerate long duration faults. The evaluation results showed that the presented system reduces the execution time of error detectors to 4.56ns and their code size by 56%, and removes the identified time-of-check-to-time-of-use (TOCTTOU) vulnerability windows and can recover under transient and long duration faults. These high error detection and recovery coverage and low overheads in TMR-based systems are due to the design of software-based fault tolerance techniques by using fault injection data.

6. Future Work

In the future, focus will be on error detection in all the inputs at a same time. As in recent technique this facility is not available. By detecting and correcting the errors from all the three input and more important at the same bit gives accurate system. This system then after never requires any maintenance and it works for long period of time. Fault Detection and Correction System searches the state space of the system until it either finds an undetected error or exhausts the state space. This search is automatic in the sense that it does not require user guidance once the system has been modeled. However, focus will only on the write operation between then processor and the memory. In order to simplify the process description, the read operation that the processor reads data from the memory is not considered. In general, it is just an early work. In the future, work will be to investigate the read operation and apply the method to some complicated system.

References

1. Kashif Sagheer Siddiqui, Mirza Altamash Baig. FRAM based TMR (Triple Modular Redundancy) for Fault Tolerance implementation. *Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, 2005
2. Wei Chen, Rui Gong, Fang Liu, Kui Dai, Zhiying Wang. Improving the Fault Tolerance of a Computer System with Space-Time Triple Modular Redundancy. *Proceedings International Conference on Dependable Systems and Networks*, pp. 389-98, 23-26 June 2006
3. Mark Hunger and Sybille Hellebrand. The Impact of Manufacturing Defects on the Fault Tolerance of TMR-Systems. *25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, (2010)
4. Chris Winstead, Yi Luo, Eduardo Monzon, and Abiezer Tejeda. An error correction method for binary and multiple-valued logic. *41st IEEE International Symposium on Multiple-Valued Logic*, 2011.
5. Jun Yao, Ryoji Watanabe, Kazuhiro Yoshimura, Takashi Nakada, Hajime Shimada, and Yasuhiko Nakashima. An Efficient and Reliable 1.5-way Processor by Fusion of Space and Time Redundancies. *IEEE TRANSACTION*, 2011.
6. Jakob Lechner. Designing Robust GALS Circuits with Triple Modular Redundancy. *Ninth European Dependable Computing Conference*, 2012.
7. Ping-Yeh Yin et al. A Multi-Stage Fault-Tolerant Multiplier with Triple Module Redundancy (TMR) Technique. *4th International Conference on Intelligent Systems, Modelling 2013*
8. C. W. Chiou. Concurrent error detection in array multipliers for GF(2m) fields. *Electron. Lett.*, vol. 38, no. 14, pp. 688–689, Jul. 2002.
9. M. Valinataj and S. Safari. Fault tolerant arithmetic operations with multiple error detection and correction. in *Proc. IEEE Int. Symp. Defect and Fault-Tolerance in VLSI Syst.*, 2007, pp. 188–196.
10. C. Y. Lee, W. Y. Lee, and P. K. Meher. Fault-tolerant bitparallel multiplier for polynomial basis of GF(2m). in *Proc. IEEE Int. Conf. Circuits Syst. Testing and Diagnosis*, 2009, pp. 1–4.
11. D. Marienfeld, E. S. Sogomonyan, V. Ocheretnij, and M. Gossel. A new self-checking multiplier by use of a code disjoint sum-bit duplicated adder. in *Proc. Ninth IEEE European Test Symp. (ETS '04)*, 2004, pp. 30–35.
12. B. K. Kumar and P. K. Lala. On-line detection of faults in carry-select adders. in *Proc. Int'l Test Conf. 2003 (ITC '03)*, 2003, pp. 912–918.
13. D. P. Vasudevan, P. K. Lala, and J. P. Parkerson. Selfchecking carry-select adder design based on two-rail encoding. *IEEE Trans. Circuits Syst. I*, vol. 54, no. 12, pp. 2696–2705, Dec. 2007.
14. R. Forsati, K. Faez, F. Moradi, and A. Rahbar. A fault tolerant method for residue arithmetic circuits. in *Proc. IEEE Int. Conf. Information Management*.
15. Shubham C. Anjankar, M. T. Kolte. Fault Tolerant and Correction System Using Triple Modular Redundancy (TMR). Paper published in *International Journal of Emerging Engineering Research and Technology, IJEERT*, volume 2, Issue 2, May 2014.
16. G. A. Reis, J. Chang, N. Vachharajani, et al. SWIFT: Software Implemented Fault Tolerance. in *Proceedings of the International Symposium on Code Generation and Optimization*, pp. 243-254, 2005.
17. P. Akritidis, C. Cadar, C. Raiciu, et al. Preventing Memory Error Exploits with WIT. in *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.
18. S. K. Sahoo, M.-L. Li, P. Ramachandran, et al. Using Likely Program Invariants to Detect Hardware Errors. in *Proc. DSN*, pp. 70-79, 2008.
19. B. Pfarr, M. Calabrese, J. Kirkpatrick, and J. T. Malay. Exploring the Possibilities: Earth and Space Science Missions in the Context of Exploration. in *Proceedings of the Aerospace Conference*, 2006.
20. J. Chang, G. A. Reis, and D. I. August. Automatic Instruction-Level Software-Only Recovery Methods. in *Proc. DSN*, 2006.