International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 2, May 2014, PP 187-191

Fault Tolerant and Correction System Using Triple Modular Redundancy

Shubham C. Anjankar[1], Dr. Mahesh T. Kolte[2]
[1]Department of Electronics & Tele-communication, MIT College of Engineering, Pune, India
[2]Department of Electronics & Tele-communication, MIT College of Engineering, Pune, India

Abstract: *An alternative way to have a fault less system is Fault Tolerant System, Triple Modular Redundancy (TMR) is used for making a fault tolerant system.. FPGA platform used in Altera Cyclone kit and Altera Quartus software is for functional and timing simulation. This model not only detects the faulty processor but also repair the faulty bits in the faulty processor .Fault detection are done over the air means at the same time. By using this TMR model, the faulty processor is detected as well as the administrator will be able to know that fault lie in which bit of which processor. The timing simulation shows that time requires for fault detection and repair is in 8ns and it is very low.*

Keywords: *Fault tolerance, Triple-modular Redundancy, Cyclone, Quartus*.

1. Introduction

Fault tolerant is the capability of a system to cope with internal error and achieve its task correctly. The idea of fault tolerance is to boost the dependency of a system. A complementary but separate appearance for up rosining reliability is fault deterrence. Permanent and inseparable element in the explanation of fault tolerance is the demand or claim that there is specification of what makes up correct performance. A system collapse when a real running system diverges from this particular behavior. The reason of collapse or system crash is called an error. An error can be invalid system state, one that is not acceptable by the system behavior requirements. The error itself is the outcome of failing in the system or fault [1] - [3].

Fault is the core (basic) cause of a system failure. That means an error is only as specified and nothing more than the sign of fault. A fault might not always results in an error, but the same fault may outcome in numerous errors. Similarly a single error may raise a numerous failures [3].

On the way to clarify Triple Modular Redundancy, it is necessary to elaborate the idea of triple redundancy. The idea is shown in Figure 1. Taking advantage of FPGA, the TMR insta- ntiated inside become easily modified and upgraded in the future [3].

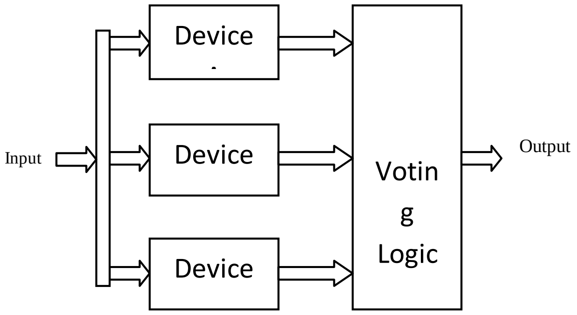


Figure 1. *Block Diagram of TMR*

Basically, a TMR system is composed of three identical devices and voting logic. The voting logic is the majority voter which takes the majority of inputs to be the output value. Since Device B and Device C are replication of Device A and they all accept the same input value, the output of A, B and C should be consistent in theory. Due to the fault in system, one of these three devices may have an error inside and generate a different output. [4]This inconsistency will be caught and corrected by voting logic .Thus; the voted output is always a correct value under the assumption of single error. Thus, the voted output is always a correct value under the assumption of single error.

When the TMR concept is applied to a processor (system), all output signal of the CPU are voted; therefore no error should exist at output of voters. Any error that occurs represent that one of the CPUs has an error inside .If that error is not corrected by some way; it may result in more errors and finally become unrecoverable. [3]- [4]

The error encode in fig. 2 is a device that will analyze error signal offered by voters and find out which CPU generates the error. Once the faulty CPU is identified, some extra circuit will interrupt all three processors and correct that error. If any one of the three system faults shots, the other two systems can correct and cover the fault. The error circuits turn high whenever any one of the output diverges from the other two.

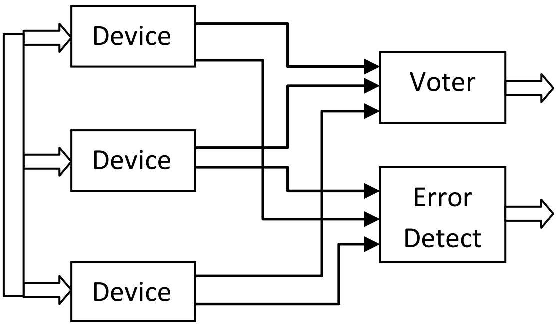


Figure 2. *System Voter*

2. Concepts of Fault Tolerance

In the design of fault-tolerant systems, the designer must consider the possible occurrence of several different kinds of faults such as transient faults, intermittent faults, permanent, logical faults, and indeterminate faults. Transient faults, often caused by external disturbances, exist for a finite length of time and are nonrecurring. Intermittent faults occur periodically and typically result from unstable device operation. Permanent faults are perpetual and can be caused by physical damage or design errors. Logical faults occur when inputs or outputs of logic gates are stuck-at-0 or stuck-at-1. Indeterminate faults occur when inputs or outputs of logic gates float between logic 0 and logic 1 [4].

A system can operate correctly in the presence of the aforementioned faults if the appropriate form of redundancy is incorporated into the system. Two major fault tolerant design approaches are static and dynamic redundancy. Static redundancy is the use of redundant components so that faults may be masked. Dynamic redundancy is the reorganization of a system so that the functions of a faulty unit are transferred to other functional units. Four specific types of redundancy are information redundancy, time redundancy, software redundancy, and hardware redundancy. Information redundancy is the use of error detecting or error correcting codes for information representation. Time redundancy is the repetition of system operations so that transient faults can be masked. Software redundancy is the inclusion of several alternative programs for system operations so that software faults (design mistakes) can be tolerated. Hardware redundancy is the inclusion of multiple copies of critical components so that inter- mittent and permanent faults can be tolerated.

Hardware redundancy is the concept used in a very popular architecture for fault-tolerant processors. A multiprocessor system is a computer system that is made up of several CPUs or, more generally, processing elements which share compu- tational tasks. Multiprocessors are different from multicomputer systems which have several proce- ssing elements working independently on separate tasks [5].

3. Impact of Soft Errors in Sequential Circuits and Combinational Circuits

The circuit of modern processor or other electronic system falls into two basic classes: seq-uential circuit and combinational circuit. Soft errors in these two circuits have different impact. Thus, different approaches are required to protect the sequential circuit and the combinational circuit.

3.1 Errors in Sequential Circuits

The main contribution to the soft error rate (SER) comes from sequential circuits in current microprocessors. Sequential circuits always refer to different storage elements, such as registers, memories, counters and flip-flops in general. A soft error in these circuits may result in a bit flip in the saved state, which may lead to a wrong execution. Storage elements take up a large part of the chip area in modern microprocessors. As a result, most modern microprocessors already incorporate mechanisms for detecting soft errors, like the triple modular redundancy technique [13].

3.2 Errors in Combinational Circuits

A particle that strikes a p-n junction within a combi- national circuit may alter the value produced by the circuit. However, a transient change in the combinational circuit will not affect the results of a computation unless it is captured by a sequential circuit. Transient changes on the clock signal or reset signal will definitely cause the circuit incorrectly executed. Past research has shown that combinational logic is much less susceptible to soft errors than memory elements [11] and the probability of the glitch from the combinational circuit captured by the sequential circuit is very small.

With the trends of reduced feature sizes, supply and threshold voltages, soft error tolerance of combinational logic circuits is affected more than memory elements. In addition, higher clock frequencies increase the chance of a glitch being captured by a sequential element [7-12]. For processors where the sequential elements have been protected, combinational logic will quickly become the dominant source of soft errors.

4. Error Detection and Correction Approach

In most of the work published to date on error correction with C-elements, a straightforward logic-case analysis is used. The C-element is a Well-known gate that has long been used in asynchronous circuit design [6], and was more recently recognized for its inherent fault compensating abilities. The C-element may be used to correct momentary faults if A and B are two redundant copies of the same logic value. Under normal conditions the two inputs should be equal and change at the same time. If a momentary error appears on only one of the signals, then the output remains unaffected. Hence the C element can correct any single momentary error using only two redundant signals. As with TMR, two simultaneous errors will cause an error to propagate at the gate's output. This case analysis helps to visualize the error correcting capabilities of the C-element, but a more precise understanding is obtained by analyzing the gate's error statistics [9-10].

Approach used in this paper is different from traditional methods. The error detection table is shown in table I. As per the error the CID_0, CID_1, D_ERR gives their value where CID stands for Change in Data.

When only single processor is faulty then Y is 0, when two processors are giving faulty out then Y turns to 1. As per the combination one CID_0, CID_1 the faulty processor is detected [5] - [7].

Error correction is also based on the value of CID_0, CID_1, Y and D_ERR. According to the combination of these four value the faulty processor and faulty bit is detected and then these processor is assign the correct value. V_ERR is the voter error detection output. V_ERR checks that voter itself is not a faulty.

5. Simulation Results

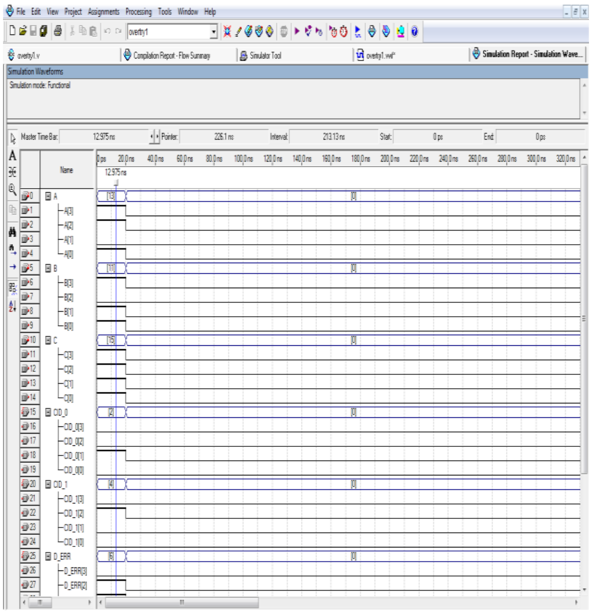
Simulation result shows that the faulty processor is detected as well as the faulty bit also notified by the timing and functional simulation. The 2nd bit of the processor A is faulty and 3rd bit of processor B also, the CID_0 is 1 and CID_1 is 0 which shows that the processor A is faulty and CID_0 is 0 and CID_1 is 1 which shows that processor B is faulty.

Table 1. *Total 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

6. Conclusion

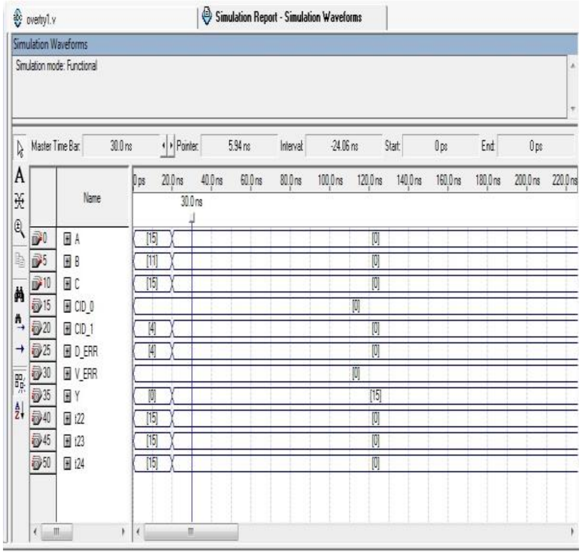
This system evaluates the processor and checks for the faulty bits in the processor. By evaluating the faults in the processor prevents the whole system from collapsing. With the help of TMR administrator will come to know which of the processor is diverting from regular program and they will able to take the appropriate action based on the results. This system not only detects he faults but also it will recover it.


Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 12.975 ns Pointer: 226.1 ns Interval: 213.13 ns Start: 0 ps End: 0 ps

Figure 3. *Error Detection*


overtry1.v

Simulation Report - Simulation Waveforms

Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 30.0 ns Pointer: 5.94 ns Interval: -24.06 ns Start: 0 ps End: 0 ps

Figure 4. *Corrected Output*

REFERENCES

[1] Kashif Sagheer Siddiqui, Mirza Altamash Baig,, "FRAM based TMR (Triple Modular Redundancy) for Fault Tole-rance 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

[5] logic", 41st IEEE International Symposium on Multiple-Valued Logic, 2011.

[6] 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.

[7] Jakob Lechner, "Designing Robust GALS Circuits with Triple Modular Redundancy", Ninth European Dependable Computing Conference, 2012.

[8] 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

[9] C. W. Chiou, "Concurrent error detection in array multipliers for GF(2m) fields," Electron. Lett., vol. 38, no. 14, pp. 688–689, Jul. 2002.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] R. Forsati, K. Faez, F. Moradi, and A. Rahbar, "A fault tolerant method for residue arithmetic circuits," in Proc. IEEE Int. Conf. Information Management

AUTHOR'S BIOGRAPHY

Shubham Chhatrapati Anjankar received the B.E. degree in Electronics and Communication with a first class from the RTM Nagpur University, Nagpur India, in June 2012, and pursuing M.E. degree in VLSI and Embedded System Engineering from the University of Pune, India.

He joined Research and Development Laboratory, VLSI and Embedded System, MITCOE, Pune, in August 2012. He completed project in Honeywell Industry in 2013.Shubham also worked with Ecosustainable Living Technology, Bangalore, India.

His research interests include fault tolerant systems, energy consumption in android, renewable energy, wind turbine inverters.

Dr. Mahesh T. Kolte is Head of Electronics and Tele-communication Department. He completed his Phd in Electronics and Tele-communication Engineering. Dr. Mahesh holds two patents in hearing aid. His research field is Signal Processing, Image Processing.