ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 2, Issue 10, October 2014

ANALYSIS OF BERGER CODE BASED FAULT TOLERANT TECHNIQUES FOR EMBEDDED SYSTEM

N.Ramkumar, Boopathy.S

Assistant Professor, Electronics and Communication Engineering, KSR Institute for Engineering and technology

Tiruchengode, Tamilnadu, India[1,2]

Abstract- With continued scaling of silicon process technology, producing reliable electronic components in extremely denser technologies pose a challenge. Further, the systems fabricated in deep sub-micron technology are prone to intermittent or transient faults, causing unidirectional errors, upon exposure to ionizing radiations during system operation. The ability to operate in the intended manner even in the presence of faults is an important objective of all electronic systems. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing run-time (or online) fault detection capabilities. Totally Self-checking (TSC) circuits permit online detection of hardware faults. The objective of this project is to analyze the area (resource utilization), speed and power consumption for n bit Look-Up Table (LUT) implementation with and without fault detection capability using Berger technique on Xilinx FPGA. The Berger code is the least redundant systematic code available for detecting all single and multi-bit unidirectional errors. This project is proposed to find the fault in 4, 8, 16, 32 bits LUT. The fault will be detected by Berger checker architecture.

Keywords: Fault Tolerance; Totally Self-Checking Circuits, Unidirectional Errors, Berger code.

I. INTRODUCTION

In recent times due to technological advancements, the performance of integrated circuits has increased a lot also lower device sizes, low power consumption have also helped to improve the performance of the devices. But on the other hand, modern devices have become more susceptible to transient faults and when these faults are executed, it creates soft errors. So soft errors are errors which are not consistent rather they are random. Although Soft errors cannot damage the physical hardware of the chip however they can corrupt the value stored in the chip. Hard errors are related to the system hardware. So the difference between soft errors and hard errors is that, soft errors can be corrected by applying different techniques where as to rectify hard errors physical changes has to be done on hardware. Soft errors can occur due to environmental conditions such as radiation flux, alpha particles, cosmic rays, power supply fluctuations, temperature, pressure, humidity and electromagnetic interference.

The errors occurring in an electronic circuit can be broadly classified as symmetric, asymmetric, and unidirectional errors. The error is symmetric if both 0 to 1 and 1 to 0 transitions occur simultaneously in a data word. If only 0 to 1 or 1 to 0 transitions are likely, and the error type is known a priori, then the errors are asymmetric. If both 0 to 1 and 1 to 0 transitions can occur in data words, but in any particular word all errors are of one type, then the errors are called unidirectional errors. The ability to operate in the intended manner even in the presence of faults is an important objective of all electronic systems. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing concurrent fault detection capabilities.

Fault detection methods can be broadly classified as Built-In Self-test, roving technique, redundancy technique, logic implications technique and error coding technique. Redundancy is based on either modular redundancy or time redundancy. In modular redundancy, the functional module is replicated two or three times. In time redundancy, the same function is performed by the same functional module more than once. Any difference in these outputs indicates a fault. It is obvious that there is an area overhead or latency overhead by two or three times when using redundancy techniques.

Faults can be detected by verifying the code with binary data. Many unidirectional error detecting codes like Parity code, Hamming code, Reed Solomon code, Berger code and Bose code are available. Berger code can detect all multiple unidirectional errors. Self-checking circuit using Berger code can have Berger encoder implemented as a sequential circuit or as a combinational circuit. The sequential circuit implementation requires more resource overhead to implement counter circuits and takes multiple clock cycles to detect the error. The combinational circuit implementation takes more hardware latency.

II. SYSTEM DESIGN

The proposed system is an analysis of fault tolerant techniques for embedded architectures with n bit LUT. In order to achieve fault-tolerance, each module of the system must be fault-tolerant by possessing run-time (or online) fault detection capabilities. Totally Self-checking (TSC) circuits permit online detection of hardware faults. The objective of this project is to analyze the area (resource utilization), speed and power consumption for Look-Up Table implementation with and without fault detection capability using Berger combinational and sequential checker circuits on Xilinx FPGA.

This project is proposed to find the fault in 4 bit LUT.The fault will be detected by Berger checker architecture and the fault will be automatically corrected. A LUT (Lookup table) is a one bit wide memory array. LUT's can be programmed and reprogrammed to change the logical function implemented. Multiplexers and LUTs are typical configurable devices, while D latches are not really configurable, although their asynchronous control signals (reset, clock) are configurable by means of multiplexers. The Berger code counts the number of 1's in the word and expresses it in binary.

Less logic is required to implement the Berger check error detection. It detects all unidirectional bit errors, i.e., if one or more ones turn into zeros it can be identified, but at the same time, zeros converted into ones cannot be identified. If the same number of bits from one to zero as from zero to one, then the error will not be detected. If the number of data bits is k, then the check bits (c) are equal to $\log_2 (k + 1)$ bits. Hence, the overhead is $\log_2((k+1)=k)$. These codes have been designed to be separable codes that are also perfect error detection codes in a completely asymmetric channel.

The proposed system consists of

- Look Up Table without fault tolerance
- Look Up Table with Berger combinational checker circuit

III. BERGER CODE

Currently, the Berger code is most commonly used in applications where a systematic code is needed. Berger code (BC) is an optimal separable code which can detect all unidirectional errors. A Berger code word of length n bits has I information bits and k check bits, where [k=log2 (I+1)], n=I+k. A code word is constructed by forming a binary number corresponding to the number of ones in the I information bits, and appending the bit-by-bit complement of the binary number as check bits to the information bits.

The check bits for Berger codes can be generated by using two different schemes. The scheme that uses the bit-by-bit complement of the binary number corresponding to the number of 1's in the information bits is known as the B1 encoding scheme. The other scheme, which uses the binary number corresponding to the number of 0's in the information bits as check bits, is identified as the B0 encoding scheme. If the number of information bits in a Berger code is $I= (2^K-1)$, $k\geq1$, then it is called a maximal length Berger code; otherwise it is known as the non-maximal length Berger code. The Berger codes are useful for encoding the information bits in digital systems because:

1. They are separable codes. No extra decoders are required to extract the information bits, when needed for processing, from the code word.
2. They detect all unidirectional errors; these are most likely to occur in digital systems.
3. They are optimal, in terms of the number of check bits required for I information bits, among all the separable codes that detect unidirectional errors.

IV. LOOK UP TABLE

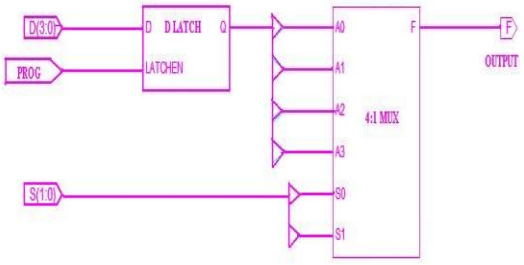

Figure 1 Block Diagram of LUT

As shown in figure 1, A LUT (Lookup table) is a one bit wide memory array. LUT's can be programmed and reprogrammed to change the logical function implemented. The fig shows the block diagram of LUT. It consists of 4 D latch and one 4:1 multiplexer. The LUT can be implemented with the cascade connection of D latch and one 4:1 multiplexer. The LUT output F depends upon the selection lines S_1 S_0. During programming, the bit stream is configured in D latch. For example: When S_1 S_0 = 00, the first D latch D0 is selected and routed to the MUX output (F). Similarly, When S_1 S_0 = 01, the second D latch D1 is selected and routed to the MUX output (F). When S_1 S_0 = 10, the third D latch D2 is selected and routed to the MUX output (F). When S_1 S_0 = 11, the fourth D latch D3 is selected and routed to the MUX output (F). When PROG signal is low, the content of D latch is configured to the output F. The function table explains the operation of LUT as shown in table 1

Control Signal	**Selection lines**	**Output**
PROG	**S_1 S_0**	**F**
0	00	D_0
0	01	D_1

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

0	10	D_2
0	11	D_3
1	xx	Z

Table 1 Function Table for LUT

V. LOOK UP TABLE WITH BERGER CHECKER CIRCUIT

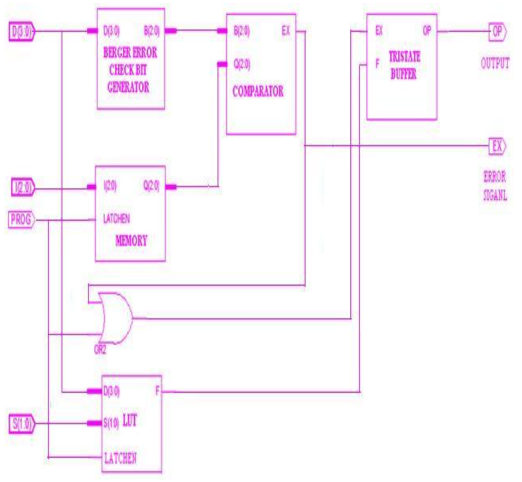

Figure 2 Block Diagram of Look Up Table With Berger Checker Circuit

As shown in figure 2, the block diagram consists of Look up Table (LUT), memory, Berger Error check bit generator, Comparator and Tristate buffer. A LUT is a memory array which can be reconfigured and consists of four D latches and 4:1 multiplexer. The Berger error checker structure consists of full adder and two half adder modules. It is available for detecting the single and multi-bit errors. The other block is memory which consisting of three D latches. The predefined bit stream is stored in memory. The common four inputs D (3:0) are given to LUT and Berger error check bit generator. If PROG is enabled, then the LUT start to store the input value else disabled, then the process is enabled in the overall blocks.

The information is being stored in the LUT. D latches and 4:1 multiplexer are used to process that information through selective lines S1 and S0 to generate the output F. The contents of D latches are given to the Berger error check bit generator to count the number of 1's in the bit stream. The output of Berger error check bit generator is compared with the predefined bit stream from memory element. LUT output and comparator output are given to Tristate buffer. If comparator produces an error output the Tristate buffer goes to high impedance else no error in the comparator, the LUT output will be the tri state buffer output.

For example, D has the value of '0001', D is the input for LUT and Berger error check bit generator. The output of LUT will be defined using select lines through multiplexer. The Berger error check bit generator will convert '0001' to '001' where counting the number of 1's in input bit stream. The memory element already has the predefined value '001'.The comparator compares the both output values from memory element and Berger error check bit generator. If 0 is output, then the LUT output is the output of Tristate buffer else 1 is output, Tristate buffer goes to high impedance state. If PROG is disabled, then the output will be available in tri-state buffer output as shown in table 2

LUT	**Berger checker circuit**	**Memory element**	**Comparator output**	**Output**
0001	001	001	0	LUT output
0000	000	001	1	High impedance

Table 2 Function Table for Block Diagram of Look Up Table With Berger Checker Circuit

VI. BERGER ERROR CHECK BIT GENERATOR

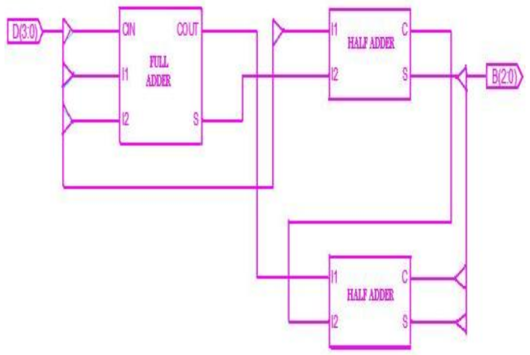

Figure 3 Block Diagram of Berger Error Check Bit Generator

As shown in figure 3, The Berger error check bit generator generates a 3 bit information by counting the number of 1's in the stored input bit stream using a full adder followed with a half adder1. The sum of the half adder1 is the LSB and the carryout of full adder and the half adder is provided to another half adder2 to generate the remaining two bits with its carry as MSB.

VII. SOFTWARE DESCRIPTION XILINX ISE

Xilinx Integrated Software Environment is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

SYNTHESIS AND SIMULATION DESIGN FOR XILINX

The Synthesis and Simulation Design Guide provides a general overview of designing Field Programmable Gate Arrays (FPGA) devices with Hardware Description Languages (HDLs). It includes design hints for the novice HDL user, as well as for the experienced user who is designing FPGA devices for the first time. Before using the Synthesis and Simulation Design Guide, you should be familiar with the operations that are common to all Xilinx tools.

IV. HARDWARE DESCRIPTION

The Spartan-3 families of Field-Programmable Gate Arrays are specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. Spartan-3 FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection and digital television equipment.

The Spartan 3 Development Kit provides a platform for engineers designing with the Xilinx Spartan 3 FPGA. The board provides the necessary hardware to not only evaluate the features of the Spartan 3 but also to implement complete user applications.

The Spartan-3 Development board was designed to support the Spartan-3 FPGA in the 676-pin, BGA package (FG676). The FG676 package supports three mid-range densities (1000, 1500, and 2000). The board was designed to support two of the three densities: the 3S1500 and 3S2000. The schematic symbol used for the Spartan-3 device indicates the specific I/O pins available in each density (396 I/O's with 2VP7 and 556 I/O's with the 2VP20/30). The Spartan-3 Development board supports Boundary-scan as well as Master/Slave Serial and Master/Slave Parallel (Select MAP) using the on-board PROMs.

The Platform Flash PROM(s) provide easy-to-use non-volatile storage for the configuration file. These devices are in–system programmable via the boundary scan chain and may program the FPGA in Master Serial, Master Select MAP, Slave Serial, or Slave Select MAP modes. The Spartan-3 Dev Board includes two standard 6-pin Mini-Din (PS2) connectors labeled JS1 and JS2.

The Spartan-3 Development board is populated with 32MB DDR SDRAM, 16MB Flash, and 2MB SRAM. Additional memory including Flash, SDRAM, and SRAM are available. The Spartan-3 Development Kit includes a 5V AC/DC Adapter that plugs into the board at "J7". The Spartan-3 Development board uses a 5V AC/DC adapter (supplied with the kit) with center positive barrel connector.

V. SIMULATION RESULTS

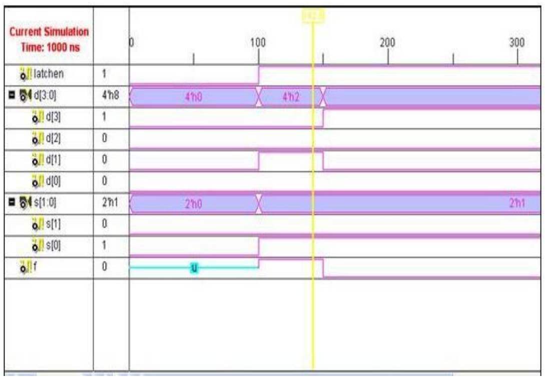

Figure 4 Simulation output for LUT

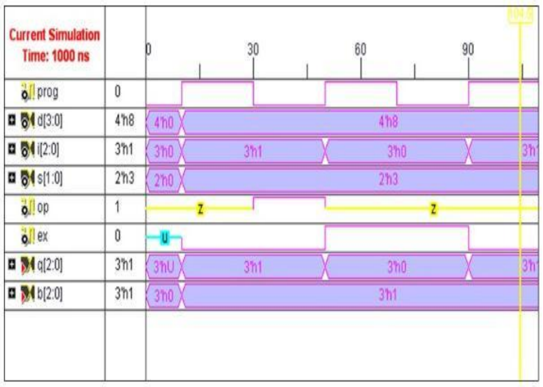

Figure 5 Simulation output for LUT with Berger combinational checker circuit

VI. IMPLEMENTATION RESULT:

Parameters	**LUT without Fault Tolerance**		**LUT with Berger combinational checker circuit**	
Device Utilization				
Logic Utilization	**Used**	**Utilization**	**Used**	**Utilization**
Number of Slices	2	0%	8	0%
Number of slice Flip	5	0%	4	0%

Flops				
Number of 4 input LUTs	4	0%	12	0%
Number of bonded IOBs	8	5%	12	8%
Number of GCLKs	2	25%	2	25%
Timing Summary				
Speed Grade	-5		-5	
Minimum period	No path found		No path found	
Minimum input arrival time before clock	2.637ns		2.832ns	
Maximum output required time after clock	6.141ns		10.311ns	
Maximum combinational path delay	No path found		11.195ns	

Table 3 Comparison summary

VII. CONCLUSION

As technology advances, the problem of soft errors is spreading widely. So, some techniques are required which can reduce the existing soft errors and increase the performance as well as reliability of a system. The project uses the technique of Berger combinational and sequential circuits on Xilinx FPGA for which can detect the soft error very efficiently and it is very less time consuming as well as cost hence the fault-tolerance, of each module of the system must be fault-tolerant by possessing run-time (or online) fault detection capabilities. Totally Self-checking (TSC) circuits permit online detection of hardware faults. Berger checker structure is used to utilize less number of LUTs.

REFERENCES

[1] Boudjit. M., M. Nicolaidis, andK.Torki, (1993)'Automatic Generation Algorithms, Experiments andComparisons of self-checking PLA schemes usingparity codes', in Proc. 4th European Conference on Design Automation, pp. 144-150.
[2] Jha.N.K, and S. J. Wang, (1993) ,'Design and Synthesis of Self-Checking VLSI Circuits' in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 6, pp. 878-887.
[3] Kavousianos. X, D. Nikolos, G. Foukarakis, T. Gnardellis (1999) 'New efficient totally self-checking Berger code checkers' Integration,the VLSI journal 28 pp 101-118.
[4] J.C. Lo et al., (1989), 'Concurrent Error Detection in Arithmetic and Logical operations using Berger Codes' Proc. 9th Symp. Comput. Arithmetic, pp. 233 – 240.
[5] Metra .C, "Novel Berger Code Checker,(1995),' IEEE Proceedings on Defect and Fault tolerance in VLSI systems', pp. 287-295.
[6] Natarajan Somasundaram, Jeong A Lee, FarhadMehdipour, Ramadass Narayanadass, Y V Ramana Rao, (2013),'Scalable Error Detection Coding© Algorithm for Totally Self-Checking (TSC) Circuits SEDC© Algorithm for TSC Circuits'Consumer Electronics Times, Vol. 2 Iss. 3, PP. 116-123.
[7] Piestrak. S. J., (1987), 'Design of Fast Self-Testing Checkers for a Class of Berger Codes', in IEEE Transactions on Computers, vol. C-36, no.5, pp. 629-634.
[8] PierceD.A. Jr. and P.K.Lala (1996) 'Modular Implementation of Efficient Self Checking Checkers for the Berger Code'J. of Electronic Testing: Theory andApplications, pp. 279 – 294.
[9] Pradhan. D. K., and J. J. Stiffler,(1980) "Error-Correcting Codes and Self-CheckingCircuits", in Computer, vol. 13, no. 3, pp. 27-37.
[10] Srinivasan,V, J. W. Farquharson, Student Member, IEEE, W. H. Robinson, Member, IEEE, B. L. Bhuva, Senior Member, IEEE ,(2006)'Evaluation of Error Detection Strategies for an FPGA-Based Self-Checking Arithmetic and Logic Unit'.
[11] Stott.E, P. Sedcole, and P. Cheung, (2008),'Fault tolerant methods for reliability in FPGAs', in Proc. International Conference on Field Programmable Logic (FPL), pp.415-420.
[12] Tao. D. L., P. K. Lala, and C. R. P. Hartmann, (1988)'A MOS Implementation of Totally Self-checking Checker for the 1-out-of-3 Code', in IEEE Journal of Solid State Circuits, vol. 23, no. 3, pp. 875-877.

BIOGRAPHIES

Ramkumar.N working as an Assistant Professor in ECE department at K S R Institute for Engineering & Technology, Tiruchengode. He received his M.E in Applied Electronics from KS Rangasamy College of Technology, Tiruchengode in 2010. He pursued his B.E in Electronics & Communication Engineering K S R College of Technology, Tiruchengode in 2006. He had published two papers in national journals. He also got two proceedings in national conferences. He had 6 years of teaching experience in the field of signal processing and VLSI. His research interest includes bio medical and VLSI signal processing.

Boopathy.S working as an Assistant Professor in ECE department at K S R Institute for Engineering & Technology, Tiruchengode. He received his M.E in Embedded System Technologies from Regional Center, Anna University Coimbatore in 2014. He pursued his B.E in Electronics & Communication Engineering from

Bannari Amman Institute of Technology, Sathyamangalam, Tamilnadu, India in 2012 with distinction. He had published two papers in international journals. He also got two proceedings in international & national conferences.