International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 7, No. 2, July 2018, pp. 74~81
ISSN: 2089-4864, DOI: 10.11591/ijres.v7.i2.pp74-81

Berger Code Based Concurrent Online Self-testing of Embedded Processors

G. Prasad Acharya[1], M. Asha Rani[2]
[1]Department of ECE, Sreenidhi Institute of Science and Technology, Hyderabad, TG, India.
[2]Department of ECE, JNTUH College of Engineering, JNTUH University, Hyderabad, TG, India

Article Info

Article history:

Received March 16, 2018
Revised May 17, 2018
Accepted May 31, 2018

Keywords:

Berger Code
DLX RISC
Online testing
Single event upset

ABSTRACT

In this paper, we propose an approach to detect the temporary faults induced by an environmental phenomenon called single event upset (SEU). Berger code based self-checking checkers provides an online detection of faults in digital circuits as well as in memory arrays. In this work, a concurrent Berger code based online self- testable methodology is proposed and integrated in 32-bit DLX Reduced Instruction Set Computer (RISC) processor on a single silicon chip. The proposed methodology is implemented and verified for various arithmetic and logical operations of the DLX processor. The FPGA implementation of the proposed design shows that a meager increase in hardware utilization facilitates online self- testing to detect temporary faults.

Corresponding Author:

G. Prasad Acharya,
Department of Electronics and Communication Engineering,
Sreenidhi Institute of Science and Technology, Hyderabad, TG, India.
Email: gprasad04@gmail.com

1. INTRODUCTION

The transistor miniaturization and integration density rate in today's VLSI technology has been following the Moore's law and even More than Moore's law. Since early 2000, the VLSI technology has been shrunk into a deep submicron technology which has led to the development of complete system in a single silicon chip and the technology has been named as System-on-chip (SoC) technology. Today's Multi-processor System on-chip (MPSoC) and Network on-chip (NoC) technologies with high level integration are offering either server-based or cloud-based massively parallel processing. Graphics Processing Unit (GPU) processors developed by NVIDIA consist of massively parallel core (even up to thousands) architecture that are capable of executing thousands of smaller tasks simultaneously thereby increasing the computational speed by many folds. These processors play an important role in accelerating the computational speed in very high data involved applications such as artificial intelligence in automobiles, drones and video surveillances.

RISC based processors are the backbones of application specific embedded systems. RISC provides a platform wherein a small-set of instructions are made available for specific tasks so that the execution takes place at much higher speed i.e. even more than millions of instructions per second, hence also called as MIPS technology.

Due to sub-micron miniaturization and high integration density of transistors, today's ICs are becoming more and more susceptible not only to manufacturing defects but also to the environmental disturbances such SEU. The manufacturing defects, a majority of them being stuck-at faults may cause the permanent failure of the system and are well targeted with offline-Built in Self-test (BIST) methodology [1], [2]. The temporary faults are harder to detect during test because these faults are unlikely to occur during test. These faults normally occur at field and may cause the malfunctioning of the system during the time when they occur. The online self-test methodologies have the capability of detecting such type of temporary faults without system downtime.

This paper is organized as follows: The section-2 outlines the architecture and instruction format of DLX RISC processor. The embedded processor testing methodologies are presented in section 3. The proposed concurrent online self-test methodology is presented in section 4. Section 5 discusses about the experimental work presented in this paper. Finally, the concluding remarks are presented in section 6.

2. DLX RISC PROCESSOR ARCHITECTURE

The DLX is a 32-bit Reduced Instruction Set Computer (RISC) developed based on load-store and millions of instructions per second (MIPS) architecture [3]. The DLX RISC processor is the simplest architecture (as shown in Figure 1) used for academic purpose and is the basic architecture for commercially available RISC processors. The DLX architecture includes a register set of 32 registers each of size 32-bits wide and a 32-bit Program counter (PC). The processor is based on five-stage pipelining architecture. These pipeline stages are Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM) and Write back (WB).

During IF stage, a 32-bit instruction will get fetched from memory. The PC holds the address of next instruction to be fetched (i) by incrementing PC by 4 in case of sequential execution and (ii) branch target address predicted by the branch prediction logic. During ID stage, the instruction decoder decodes the 32-bit instructions into various fields as given in Table 1 and determines the required operands and branching address. During EX stage, the ALU performs the arithmetic and logical operations on the operands decoded/provided by the instruction decoder. During MEM stage, the computed results will be stored in the data memory. The result will be written back into register during WB stage. The MEM and WB cycles can be performed in a single clock cycle and hence the execution of an instruction can be completed in 4 clock cycles.

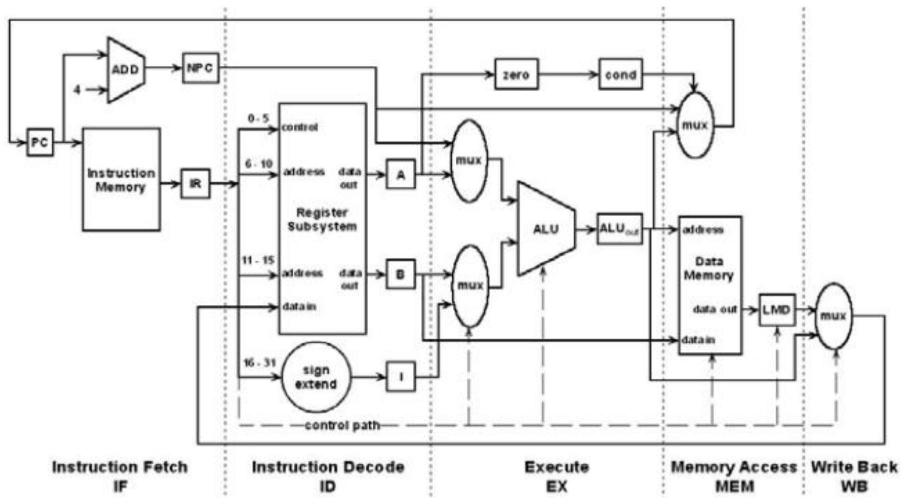


Figure 1. Architecture of DLX RISC processor

Clock pulse (n) $\longrightarrow$

1	2	3	4	5	6	7	8
IF1	ID1	EX1	MEM1	WB1			
	IF2	ID2	EX2	MEM2	WB2		
		IF3	ID3	EX3	MEM3	WB3	
			IF4	ID4	EX4	MEM4	WB4

Figure 2. Five stage pipelining of DLX processor

The Arithmetic and Logic Unit (ALU) performs the 32-bit integer and floating point (single and double precision) arithmetic operations and logical operations. All the instructions in DLX processors are

32-bit long and can be divided into the following three classes according to the type of the operation: R (*register*)-type, I (*immediate*) -type and J (*jump*)-type. In R-type instructions, three registers (two source registers and one destination) are specified in the instruction. In I-type instructions, one source register, 16-bit immediate operand (sign extended to 32-bit) are used. The J-type instructions consist of 6-bit opcode and 26-bit operand. The destination address is calculated using the 26-bit operand value. Table 1 summarizes the instruction formats of DLX processor.

Table 1. Instruction formats of DLX RISC

Instruction	Bits				
Type	31-26	25-21	20-16	15-11	10-0
R-type	0x0	R1	R2	Rd	Unused
I-type	Opcode	R1	R2	Immediate	
J-type	Opcode	Value			

3. OVERVIEW OF EMBEDDED PROCESSOR TESTING

The digital circuit testing techniques can be broadly classified into external testing and self-testing. The conventional method of manufacturing test of digital circuits is carried out using Automatic Test Equipment (ATE) hardware. The quality test patterns are generated using test pattern generation algorithm and their expected responses are stored in ATE memory.

The hardware based self-testing (also known as Built in Self-Test) of a processor facilitates the generation of test patterns using LFSR, application of the test patterns to the processor under test and the analysis of test responses for its functional correctness without the use of any external circuitry. The processor uses the internal resources of the processor such as existing hardware, memory and other test support hardware. The major issues to be dealt carefully with the BIST is the Hardware overhead, test data generation and test application time, performance degradation especially in critical paths in case of high performance devices, power consumption during self-testing.

The software-based self-testing (SBST) [4]-[9] provides an alternative solution for the above described limitations of hardware based self-testing methodology. In this methodology, generation and application of test patterns for the processor under test and response analysis are carried out by especially written software routines executed on the processor itself. The self-test routine is stored in instruction memory and the data needed is stored in Data memory of the processor. A simplified processor model for software based self-testing is shown in Figure 3.

The SBST is based on Instruction Set Architecture (ISA) and the Register Transfer Language (RTL) description of the processor and the test engineer need not have complete hardware and the details of structural fault model. The processor executes the test programs at its actual speed and hence SBST is capable of providing at-speed test solutions unlike to hardware-based BIST. However SBST is capable of providing at-speed self-test solutions to the processor for functionality verification both at manufacturing and/or field level, it cannot substitute the structure based test approaches like BIST and hence can be used to supplement the structural based test approaches to provide a more quality test.

Most of the BIST approaches (either hardware or software based) found in the literatures are off-line or non-concurrent test approaches. In these approaches either the functionality of the processor is suspended or the processor is in idle mode during test and test is not carried out concurrently with its functional operation. In concurrent online Self-test approach which is the main contribution of this paper, both the functional operation and test operation will be carried out simultaneously.

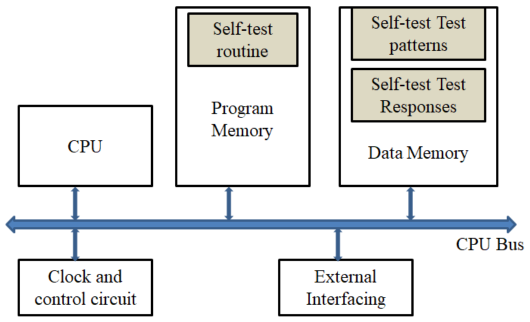


Figure 3. Processor model for software-based self-testing

4. PROPOSED ONLINE SELF-TESTABLE METHODOLOGY

The major contribution for malfunctioning of digital circuits/systems in the field (while on operation) is due to the temporary (dynamic) faults [10], [11]. These faults may be caused by radiation and other hard environmental conditions. The single event upset (SEU) is the radiation-induced errors in microelectronic circuits that may change the behavior of dynamic circuits as well as memory devices. Since these faults are non-recurrent and harder to detect during test using offline BIST, online self-test methodologies are capable of detecting the temporary faults and hence its importance [3], [12]. This paper presents the design of Berger code based totally self-checking checkers (TSC) [13], [14] to detect both permanent stuck-at faults as well as temporary faults in the DLX RISC processor. Figure 4 shows the generalized architecture for the proposed self-testable processor.

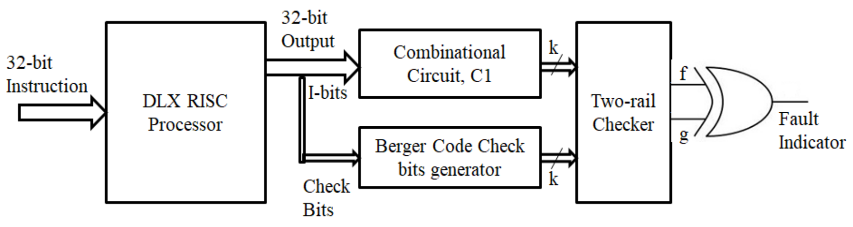


Figure 4. Simplified architecture of proposed self-testable DLX processor

4.1. Totally Self-Checking Checkers (TSC) Using Berger Code

A Berger code forms a unidirectional error detecting code where check bits represents the number of zeros in the information bit sequence, I. The number of check bits (k) for the information sequence of length n bits is evaluated using the inequality$k = \lceil \log_2^{(n+1)} \rceil$. The combinational circuit, C1 in Figure 4 is a combinational circuit that produces the complement of the check bits which is then fed to 2-rail checker along with the k-check bits. The two rail-checker produces two complementary outputs f and g in no fault case otherwise it produces f and g identical.

4.1.1. Combinational Circuit, C1

A Berger code is said to be maximal length Berger code has if n= (2^k-1) otherwise it is non-maximal length Berger code. The combinational circuit C1 in Figure 4 produces an output which is the binary equivalent of the number of 1's in the information sequence, I. In order to compute the check bits for non-maximal length Berger code, we define a number m=I0 mod (k+1), where I0 is the number of 0's in the sequence I. The Berger code check bits is the binary equivalent of m and its length is equal to $[\log_2(k+1)]$.

4.1.2. Two-rail Checker

The two-rail checker as shown in Figure 5(a), is 1-out-of-2 code which receives two groups of inputs $X=(x_1, x_2, \ldots x_n)$ and $Y=(y_1, y_2, \ldots y_n)$ from the functional circuit and produces two outputs f and g that are complement to each other. As long as $y_i=(x_i)^|$ is satisfied, the outputs of the two-rail checker will be f=0 and g=1. The totally self-checking two-rail checker can be extended for any arbitrary pairs (x_iy_i and $x_{i+1}y_{i+1}$) of inputs as given in the structure of Figure 5(b).

4.2. Berger code predictions for ALU operations

An Arithmetic and Logical Unit (ALU) is the heart of any processor and performs various arithmetic and logical operations. This section presents the predictions of Berger code for various ALU operations. Consider two n-bit operands be $A=(a_n, a_{n-1}, \ldots a_2,a_1)$ and $B=(b_n, b_{n-1}, \ldots b_2,b_1)$. Let A_c and B_c be the Berger code of A and B respectively.

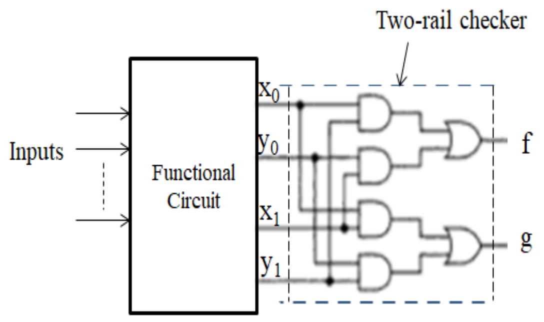


Figure 5 (a). Self-checking 2-rail checker

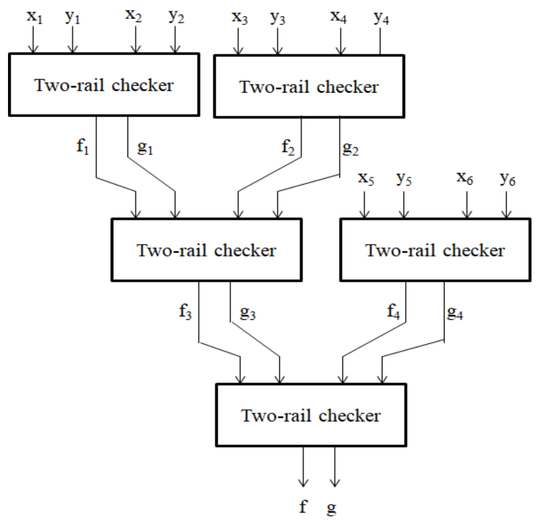


Figure 5 (b). Self-checking 2-rail checker with 6 inputs

4.2.1. Addition ($Y=A+B+c_{in}$)

The Berger code of the sum (Y_c) is computed as $Y_c=A_c+B_c-c_{in}-C_c+c_{out}$, where c_{in}, c_{out} and C_c are input carry, output carry and Berger code of intermediate carries $C=(c_n, c_{n-1}, \ldots c_2,c_1)$ respectively.

4.2.2. Subtraction

The Berger code of the difference (Y_c) is computed as $Y_c=A_c-B_c+b_{in}+BI_c-b_{out}$, where b_{in}, b_{out} and BI_c are in borrow, output borrow and Berger code of intermediate borrows $BI=(bi_n, bi_{n-1}, \ldots bi_2,bi_1)$ respectively.

4.2.3. 2's complement subtraction ($Y=A-B=A+B^{|}+1$)

The Berger code of the sum (Y_c) is computed as $Y_c=A_c-B_c-(c_{in})^{|}-N(C)+c_{out}$, where c_{in}, c_{out} and N(C) are input carry, output carry and number of 1's in the intermediate carries $C=(c_n, c_{n-1}, \ldots c_2,c_1)$ respectively.

4.2.4. Array Multiplier ($Y=A*B$)

The Berger code of the multiplier output (Y_c) is computed as $Y_c=4*A_c-4*B_c-A_c*B_c-N(C)+12$, where $N(C)=\sum_{i=1}^{m}\sum_{j=1}^{n-1}C_{i,j}$ and $C_{i,j}$ is the carry generated by the full adder in the stage of i_{th} row and j_{th} column of m-bit by n-bit array multiplier as shown in Figure 6 (m=4 and n=4).

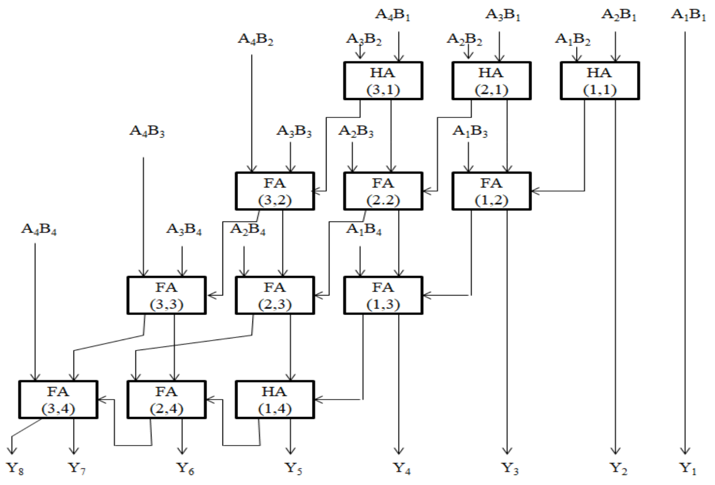

A_4B_1 A_3B_1 A_2B_1 A_1B_1
A_4B_2 A_3B_2 A_2B_2 A_1B_2
HA (3,1)
HA (2,1)
HA (1,1)
A_4B_3 A_3B_3 A_2B_3 A_1B_3
FA (3,2)
FA (2,2)
FA (1,2)
A_4B_4 A_3B_4 A_2B_4 A_1B_4
FA (3,3)
FA (2,3)
FA (1,3)
FA (3,4)
FA (2,4)
HA (1,4)
Y_8 Y_7 Y_6 Y_5 Y_4 Y_3 Y_2 Y_1

Figure 6. 4x4 binary array multiplier

4.2.5. Logical-AND (Y=A.B)

The Berger code of the Logical-AND output (Y_c) is computed as $Y_c=A_c+B_c- Z_c$ where Z_c represents the Berger code of (A|B).

4.2.6. Logical-OR (Y=A|B)

The Berger code of the Logical-OR output (Y_c) is computed as $Y_c=A_c+B_c- Z_c$ where Z_c represents the Berger code of (A.B).

4.2.7. Logical-Inverter ($Y=A^{|}$)

The Berger code of the Logical-Inverter output (Y_c) is computed as $Y_c=n-A_c$.

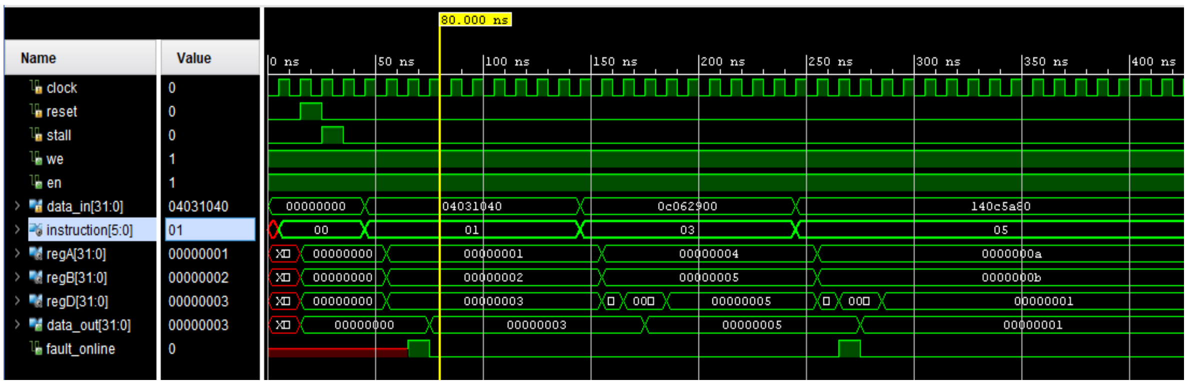
4.2.8. Logical-XOR (Y=A^B)

The Berger code of the Logical-XOR output (Y_c) is computed as $Y_c=A_c+B_c- 2*Zc+n$ where Z_c represents the Berger code of (A.B).

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

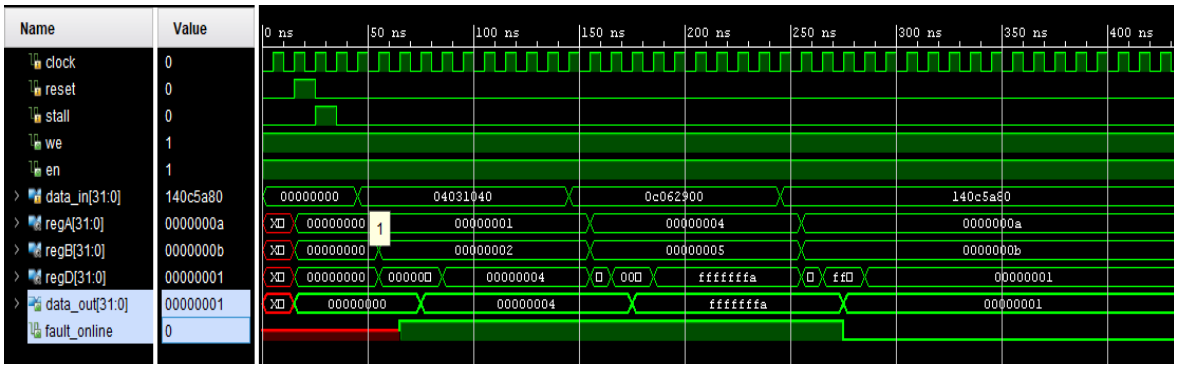
The Berger code based Totally Self-checking Checker (TSC) logic is incorporated for various arithmetic and logical operations within the RTL description of the DLX RISC processor and simulated using Xilinx Vivado 2017.2. Figure 7 demonstrates the simulation waveform of the DLX processor. The 32-bit instruction Data_in=0x04031040 is decoded as opcode=01 (ADD operation), [RA]=0x01, [RB]=0x02. The result of the ADD operation is [RD]=0x03. Similarly, Data_in=0x0C062900 is decoded as opcode=03 (Logical-OR operation), [RA]=0x04, [RB]=0x05. The result of the Logical-OR operation is [RD]=0x05. Figure 8 demonstrates the capability of Berger code based TSC to detect the presence of fault during its normal operation.

Two versions of the processor (i) standard DLX RISC architecture and (ii) TSC based Self-testable DLX RISC Processor are implemented in 7-series Zynq FPGA (xc7z020clg484-1). The design is synthesized using Xilinx Vivado synthesis tool and the synthesized netlist is shown in Figure 9. The device utilization reports are compared for both the designs as given in Table 2.



Name	Value
clock	0
reset	0
stall	0
we	1
en	1
data_in[31:0]	04031040
instruction[5:0]	01
regA[31:0]	00000001
regB[31:0]	00000002
regD[31:0]	00000003
data_out[31:0]	00000003
fault_online	0

Figure 7. Simulation results of Self-testable DLX RISC Processor (Fault-free case)



Name	Value
clock	0
reset	0
stall	0
we	1
en	1
data_in[31:0]	140c5a80
regA[31:0]	0000000a
regB[31:0]	0000000b
regD[31:0]	00000001
data_out[31:0]	00000001
fault_online	0

Figure 8. Simulation results of Self-testable DLX RISC Processor (Faulty case)

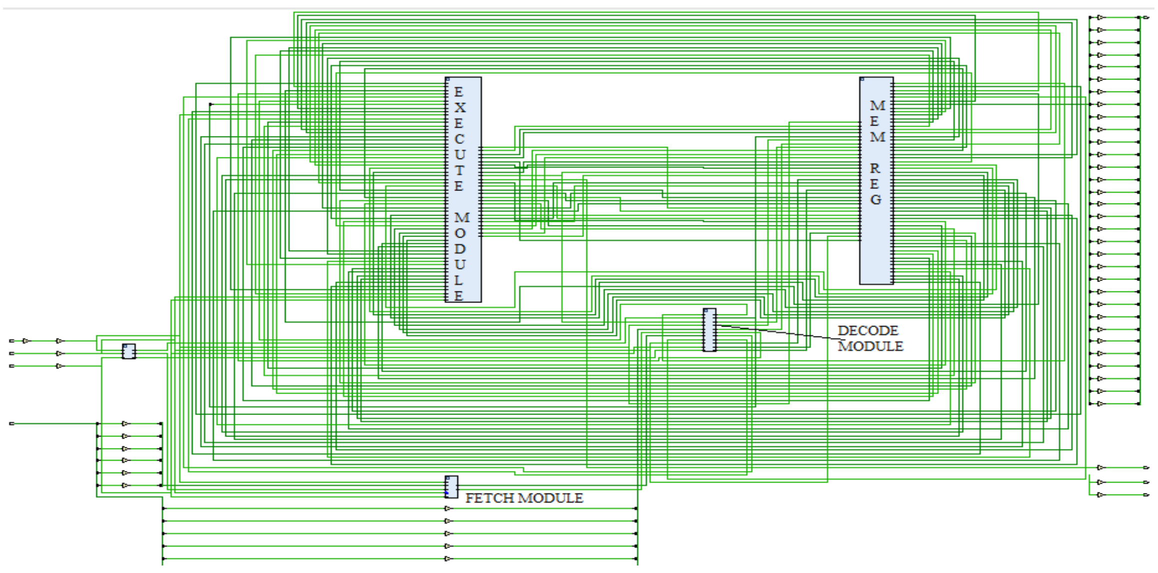


Figure 9. Synthesized net list of Self-testable DLX RISC Processor

Table 2. Comparison of Device utilization summery

Logic Resources	Standard DLX RISC Processor			TSC based Self-testable DLX RISC Processor		
	Utilized	Available	% of Utilization	Utilized	Available	% of Utilization
1. Slice Logic (LUTs)	274	53200	0.42	552	53200	1.04
2. LUT as Logic	226	53200	0.42	504	53200	0.95
3. LUT as memory	48	17400	0.28	48	17400	0.28
4. Slice Registers as FFs	155	106400	0.15	157	106400	0.15
5. DSPs	3	220	1.36	3	220	1.36
No. of Bonded IOBs	58	200	29	59	299	29.50
Total power consumption		34.527 W			38.115 W	

6. CONCLUSION

Berger code provides a unidirectional error detecting capability of detecting single or multi-bit errors in a given information sequence. Berger code prediction for the various arithmetic and logical operations has been summarized in this paper. The Berger code based totally self-checking checker (TSC) combined with two-rail checker provides a solution for the detection of temporary faults which are mainly induced by single event upset (SEU). The work presented in this work has demonstrated the concurrent online self-testing capability of DLX RISC processor. The implementation results obtained in this work has shown that the built-in self-testing capability can be incorporated in the processor design with meager overhead in the hardware (LUTs) and marginal increased power consumption.

REFERENCES

[1] Shekhar Borkar, *Microarchitecture and Design Challenges for Gigascale Integration*, Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture, p.3-3, December 04-08, 2004, Portland, Oregon.
[2] Cristian Constantinescu, Trends and Challenges in VLSI Circuit Reliability, *IEEE Micro*, v.23 n.4, p.14-19, July 2003.
[3] Hala ElAarag, A complete design of a RISC processor for pedagogical purposes, *Journal of Computing Sciences in Colleges*, Volume 25 Issue 2, December 2009 Pages 205-213.
[4] Mihalis Psarakis and Dimitris Gizopoulos, Ernesto Sanchez and Matteo Sonza Reorda, Microprocessor Software-Based Self-Testing, *IEEE Design & Test of Computers, IEEE*, 2010, Pg. 4-18.
[5] C.H.-P. Wen, L.-C. Wang, and K.-T. Cheng, ‘‘Simulation-Based Functional Test Generation for Embedded Processors,’’ *IEEE Trans. Computers*, vol. 55, no. 11, 2006, pp. 1335-1343.
[6] N. Kranitis et al., “Software-Based Self-Testing of Embedded Processors,” *IEEE Trans. Computers*, vol. 54, no. 4, 2005, pp. 461-475.
[7] D. Gizopoulos et al., ‘‘Systematic Software-Based Self- Test for Pipelined Processors,’’ *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 11, 2008, pp. 1441-1453.

[8] K. Batcher and C. Papachristou, “*Instruction Randomization Self-Test for Processor Cores*,” Proc. 17th IEEE VLSI Test Symp. (VTS 99), IEEE CS Press, 1999, pp. 34-40.
[9] By Dimitris Gizopoulos, A. Paschalis, Yervant Zorian, Embedded Processor-Based Self-Test, Kluwer Academic Publishers, 2013.
[10] C. Constantinescu. *Intermittent faults in VLSI circuits*. In Proceedings of the IEEE Workshop on Silicon Errors in Logic-System Effects, 2007.
[11] Douglas M. Blough, Gregory F. Sullivan, Gerald M. Masson, Intermittent Fault Diagnosis in Multiprocessor Systems, *IEEE Transactions on Computers*, v.41 n.11, p.1430-1441, November 1992
[12] Jiem-Chung Lo, Suchai Thanawastien, T. R. N. Rao, *Cooncurrent Error Correction in Arithemetic and Logical operations Using Berger Codes*, Proceedings of the 9th Symposium on Computer Arithmetic, Santa Monica, CA, September 1989, pp. 233-240.
[13] Parag K. Lala, Fault tolerant and fault testable hardware design, Prentice-Hall International, 1985.
[14] Parag K. Lala, Self-checking and Fault-tolerant Digital Design, Morgan Kaufmann, 2001.

BIOGRAPHIES OF AUTHORS

G. Prasad Acharya received the B. Tech (ECE) degree in the year 2004 from JNT University, Hyderabad, A. P., India. He had obtained his Masters degree in Engineering (Digital systems) from Osmania University, Hyderabad. He is currently working towards his PhD degree in the Department of Electronics and Communication at JNTUH University, Hyderabad. His area of research interests includes VLSI Design & Testing and Embedded systems. He has 13 years of teaching and research experience and is presently working as Associate Professor in the Department of ECE at Sreenidhi Institute of Science and Technology, Hyderabad, India.

Dr. M. ASHA RANI did her B.E from Osmania University, Hyderabad during 1986-90, completed her M.Tech and Ph.D from JNTU Hyderabad in 1997 and 2008 respectively. Presently she is working as Professor and Chairman, Board of Studies in the Department. of Electronics and Communication Engineering, JNTUH College of Engineering, Hyderabad. She has 27 years of teaching and research experience. Her research interest is Fault Tolerant System Design, Design for Testability, Embedded Systems Design, VLSI Design.