Berger code based concurrent online self-testing of embedded processors

G. Prasad Acharya1, † and M. Asha Rani2

1Department of ECE, Sree Nidhi Institute of Science and Technology, Hyderabad, TG, India
2Department of ECE, JNTUH College of Engineering, JNTUH University, Hyderabad, TG, India

Abstract: 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 architecture is proposed and integrated in 32-bit DLX reduced instruction set computer (RISC) processor on a single silicon chip. The proposed concurrent test 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.

Key words: single event upset; Berger code; DLX RISC; online testing

DOI: 10.1088/1674-4926/39/11/115001 EEACC: 1265A; 2220

1. Introduction

The transistor miniaturization and integration density in today’s VLSI technology is increasing at the rate predicted by Moore’s law and even at higher rates. Today’s multi-processor system on-chip (MPSoC), network on-chip (NoC) and graphics processing unit (GPU) technologies with high level integration of processing elements/cores are offering either server-based or cloud-based massively parallel processing. These processors play an important role in accelerating the computational speed in massively high data involved applications such as artificial intelligence in automobiles, drones and video surveillances. The SoC technology allows the integration of one or more processing cores (processors), embedded memory IPs and input/output (I/O) peripherals. 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.

The SoC architectures also include analog and mixed signal interfaces (analog-to-digital and digital-to-analog converters) that provide the interface between analog data acquisition units and digital processors. According to a survey[1], around 80% circuitry in SoCs is digital but 80% of faults occur in analog circuitry. Though the BIST techniques are basically developed for the detection of faults based on voltage-level based testing in digital circuits, these techniques are further developed to incorporate parametric testing for self-testing of analog circuits as well.

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 as single event upset (SEU). The SEU is a natural phenomenon wherein high energy particles like alpha and beta particles may fall onto the ICs and cause malfunctioning of the system due to the induction of temporary faults[2–6]. These faults are harder to detect during testing because these faults may not occur during test. On-line self-test methodologies available in many literatures are capable of detecting such types of temporary faults without system downtime.

This paper is organized as follows: Section 2 outlines the architecture and instruction format of the 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 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 microprocessors without interleaving pipelining system (MIPS) architecture[7]. The DLX RISC processor is the simplest architecture (as shown in Fig. 1) used for academic purposes 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 a five-stage pipelining architecture. These pipeline stages are instruction fetch (IF), instruction decode (ID), execute (EX), memory access (MEM) and write back (WB).

During the IF stage, a 32-bit instruction will get fetched from the memory. The PC holds the address of the next instruction to be fetched (i) by incrementing PC by 4 in case of sequential execution and (ii) the branch target address predicted by the branch prediction logic. During the 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 the EXE stage, the arith-

† Corresponding author. Email: gprasad04@gmail.com
Received 13 January 2018, revised manuscript received 8 May 2018 ©2018 Chinese Institute of Electronics
metic logic unit (ALU) performs the arithmetic and logical operations on the operands decoded/provided by the instruction decoder. During the MEM stage, the computed results will be written back to the data memory. The result will be written back into register during the 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.

The ALU performs 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 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 instructions. In I-type instructions, one source register and 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 format of the DLX processor.

### 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 testing the manufacturing defects in digital circuits is carried out using automatic test equipment (ATE) hardware. The quality test patterns are generated using algorithm based test pattern generation strategies as available in the related literatures and stored along with their expected responses in the 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 their functional correctness without the use of any external circuitry. The processor uses its internal resources such as the processor itself, the register file, instruction set, memory and other test support hardware. The major parameters to be considered while selecting the BIST strategy include: hardware overhead, test data generation and application time, performance degradation especially in critical paths in case of high performance devices and power consumption during self-testing. BIST capability is incorporated in a MIPS processor using a linear feedback shift register (LFSR), built-in logic block observer (BILBO) and concurrent BILBO (CBILBO)\[8\]. Various power saving techniques like weighted LFSR and dual speed LFSR have been presented to reduce the power consumption in the self-testable MIPS processor. The dynamic partial reconfiguration feature of FPGAs shall be utilized for self-testing of processor cores by dynamically reconfiguring the partial bit-files of the functional-mode processor and BIST-oriented processor onto the dynamic region of the FPGA\[9\].

The software-based self-testing (SBST)\[10–15\] provides an alternative solution for the above mentioned limitations of hardware based self-testing methodology. In this methodo-
The major contribution for malfunctioning of digital circuits/systems in the field (while on operation) is due to the temporary (dynamic) faults. These faults may be caused by radiation and other hard environmental conditions. The 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 a test using off-line BIST, on-line self-test methodologies are capable of detecting the temporary faults and are used to improve the reliability of the system. This paper presents the design of the Berger code based totally self-checking checkers (TSC) to detect both permanent stuck-at faults as well as temporary faults in the DLX RISC processor. Fig. 4 shows the generalized architecture for the proposed self-testable processor.

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. These faults may be caused by radiation and other hard environmental conditions. The 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 a test using off-line BIST, on-line self-test methodologies are capable of detecting the temporary faults and are used to improve the reliability of the system. This paper presents the design of the Berger code based totally self-checking checkers (TSC) to detect both permanent stuck-at faults as well as temporary faults in the DLX RISC processor. Fig. 4 shows the generalized architecture for the proposed self-testable processor.

4.1. Totally self-checking checkers using berger code

Among all unidirectional error detecting (AUED) codes, the Berger code forms the least redundant and separable code\(^\text{[18, 19]}\). The Berger code is available with two encoding schemes: B0 and B1. In the B0 encoding scheme of the Berger code used in this work, the check bits represent the binary equivalent of the number of zeros in the information bit sequence, \(I\). In the B1 scheme of encoding, the check bits represent the 1’s complement of the number of 1’s in \(I\). 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 Fig. 4 is a combinational circuit that produces the complement of the check bits, which is then fed to the 2-rail checker along with the \(k\)-check bits. The two rail-checker produces two complementary outputs \(f\) and \(g\) in the no fault case otherwise it produces identical values for \(f\) and \(g\).

**Combination circuit, C1**

A Berger code is said to be a maximal length Berger code has if \(n = (2^k - 1)\) otherwise it is a non-maximal length Berger code. The combinational circuit C1 as shown in Fig. 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 the non-maximal length Berger code, we define a number \(m = l_0 \mod (k + 1)\), where \(l_0\) is the number of 0’s in the sequence \(I\). The Berger code check bits are the binary equivalent of \(m\) and its length is equal to \(\lceil \log_2 (k + 1) \rceil\).

**Two-rail checker**

The two-rail checker as shown in Fig. 5(a), is a 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 complementary to each other. As long as \(y_1 = (x_1)\) 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 \((x_i, y_i)\) of inputs as given in the structure of Fig. 5(b).

4.2. Berger code predictions for ALU operations

The ALU is the heart of any processor and performs various arithmetic and logical operations. This section presents the predictions of the Berger code for various ALU operations. Consider two \(n\)-bit operands \(A = (a_{n-1}, a_{n-2}, \ldots, a_2, a_1)\) and \(B = (b_{n-1}, b_{n-2}, \ldots, b_2, b_1)\). Let \(A_c\) and \(B_c\) be the Berger code of \(A\) and \(B\) respectively.

**Addition** \((Y = A + B + c_{in})\)

The Berger code of the sum \((Y_c)\) is computed as \(Y_c = A_c + B_c\)
The Berger code of the logical-OR output \( Y = A \lor B \) is computed as \( Y = A + B + 1 \).

Subtraction

The Berger code of the difference \( Y_c = A_c - B_c \) is computed as \( Y = A_c - B_c + B_{\text{out}} \) where \( B_{\text{out}} \) is the borrow generated by the \( J \)-th sum in the \( I \)-th row. The carry \( C_{I,J} \) and the \( J \)-th row are the carry generated by the full adder in the stage of the \( J \)-th sum and the \( I \)-th row. The \( C_{I,J} \) is the carry generated by the \( J \)-th row and the \( I \)-th row. The \( C_{I,J} \) is the carry generated by the \( J \)-th row and the \( I \)-th row. The \( C_{I,J} \) is the carry generated by the \( J \)-th row and the \( I \)-th row. The \( C_{I,J} \) is the carry generated by the \( J \)-th row and the \( I \)-th row.

Array Multiplier \( Y = AB \)

The Berger code of the multiplier output \( Y_c \) is computed as \( Y = A_c \cdot B_c + c_{\text{out}} \) where \( c_{\text{out}} \) is the output carry and the Berger code of intermediate carries \( C = (c_{n-1}, c_0, c_1) \) respectively.

Logical-Inverter \( Y = \overline{A} \)

The Berger code of the logical-Inverter output \( Y_c \) is computed as \( Y = n - A_c \).

Logical-XOR \( Y = A \oplus B \)

The Berger code of the logical-XOR output \( Y_c \) is computed as \( Y = A_c \oplus B_c - 2Z_c + n \) 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. Fig. 7 demonstrates the simulation waveform of the DLX processor.

To demonstrate the self-checking capability of the implemented processor, faults are injected in the design during simulation. The Stuck-at-1 (SA1) fault is injected in the \( c_{\text{in}} \) line by forcing \( c_{\text{in}} = 1 \), which produced the output result as \( [R_{1}] = 0x04 \) as demonstrated in Fig. 8, instead of \( [R_{1}] = 0x03 \) for the same instruction Data_in = 0x04031040 shown in Fig. 7. Similarly, bit-flip of the 27th bit and the 28th bit of Data_in
= 0x0C062900 will be decoded as opcode = 05 (logical-XNOR operation), [Rₐ] = 0x04, [Rₜ] = 0x05 which produces the output result as [Rₐ] = 0xFFFFFFFFFA. The processor has produced an erroneous output in both the cases, which is indicated by fault_online = 1. These faults may be caused either by permanent defects or by SEU induced temporary faults or by any other kind of soft errors.

Two versions of the processor (i) standard DLX RISC architecture and (ii) TSC based Self-testable DLX RISC Processor are synthesized and implemented in 7-series Zynq FPGA (xc7z020clg484-1). The device utilization reports and overall power consumption for the two designs is summarized in Table 2. The last column in the table shows the hardware/power overhead required for the design (2), which can be traded-off with its ability to facilitate on-line concurrent self-testing.

### 6. Conclusion

The 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 that are carried out by a processor ALU has been summarized in this paper. The Berger code based totally self-checking checker (TSC) combined with a two-rail checker provides a solution for the online detection of SEU induced temporary faults and soft errors. The work presented in this paper has demonstrated the concurrent self-testing capability of the DLX RISC processor. The implementation results obtained in this work

| Logic resources     | Design 1 (Standard DLX RISC processor) | Design 2 (TSC based self-testable DLX RISC processor) | Percentage of overhead |
|---------------------|----------------------------------------|------------------------------------------------------|------------------------|
|                     | Utilized | Available | Percentage of utilization | Utilized | Available | Percentage of utilization |                     |
| 1. Slice logic (LUTs) | 274      | 53 200    | 0.42                     | 552      | 53 200    | 1.04                     | 148                   |
| 2. LUT as logic     | 226      | 53 200    | 0.42                     | 504      | 53 200    | 0.95                     | 126                   |
| 3. LUT as memory    | 48       | 17 400    | 0.28                     | 48       | 17 400    | 0.28                     | 0                     |
| 4. Slice registers as FFs | 155    | 106 400   | 0.15                     | 157      | 106 400   | 0.15                     | 0                     |
| 5. DSPs             | 3        | 220       | 1.36                     | 3        | 220       | 1.36                     | 0                     |
| 6. No. of bonded IOBs | 58      | 200       | 29                       | 59       | 299       | 29.50                    | 2                     |
| 7. Total power consumption | 34.527 W |            |                          | 38.115 W |            |                          | 10                    |

![Fig. 6. 4 × 4 binary array multiplier.](image)

![Fig. 7. (Color online) Simulation results of Self-testable DLX RISC Processor (Fault-free case).](image)

![Fig. 8. (Color online) Simulation results of self-testable DLX RISC processor (faulty case).](image)
show that the concurrent built-in self-testing capability can be incorporated in the processor design with meager overheads in the hardware (LUTs) and marginally increased power consumption.

References

[1] Binu D, Kariyappa B S. A survey on fault diagnosis of analog circuits: taxonomy and state of the art. Int J Electron Commun, 2017, 73: 68

[2] Constantinescu C. Intermittent faults in VLSI circuits. Proceedings of the IEEE Workshop on Silicon Errors in Logic - System Effects, 2007

[3] Blough D M, Sullivan G F, Masson G M. Intermittent fault diagnosis in multiprocessor systems. IEEE Trans Comput, 1992, 41(11): 1430

[4] Borkar S. Microarchitecture and design challenges for gigascale integration. Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture, Portland, Oregon, 2004

[5] Constantinescu C. Trends and challenges in VLSI circuit reliability. IEEE Micro, 2003, 23(4): 14

[6] Contant O, Lafortune S D. Diagnosis of intermittent faults. Discrete Event Dynamic Systems, 2004, 14(2): 171

[7] ElAarag H. A complete design of a RISC processor for pedagogical purposes. J Comput Sci Colleg, 2009, 25(2): 205

[8] Ravi S, Joseph M. FPGA implementation of low power self testable MIPS processor. Am-Eur J Sci Res, 2017, 12(3): 135

[9] Acharya G P, Rani M A. Design of online self-testable multicore system using dynamic partial reconfiguration of FPGA. Int J Appl Eng Res, 2017, 12(24): 15261

[10] Psarakis M, Gizopoulos D, Sanchez E. Microprocessor software-based self-testing. IEEE Design & Test of Computers, 2010: 4

[11] Wen C H P, Wang L C. Simulation-based functional test generation for embedded processors. IEEE Trans Comput, 2006, 55(11): 1335

[12] Kranitis N, Paschalis A, Gizopoulos D. Software-based self-testing of embedded processors. IEEE Trans Comput, 2005, 54(4): 461

[13] Psarakis M, Gizopoulos D, Hatzimihail M. Systematic software-based self-test for pipelined processors. IEEE Trans Very Large Scale Integr Syst, 2008, 16(11): 1441

[14] Batcher K, Papachristou C. Instruction randomization self-test for processor cores. Proc 17th IEEE VLSI Test Symposium, 1999: 34.

[15] Gizopoulos D, Paschalis A, Zorian Y. Embedded processor-based self-test. Kluwer Academic Publishers, 2013

[16] Lo J, Thanawastien S, Rao T R N. Concurrent error correction in arithmetic and logical operations using Berger codes. Proceedings of the 9th Symposium on Computer Arithmetic, Santa Monica, CA, 1989: 233

[17] Kavousianos X, Nikolos D, Foukarakis G. New efficient totally self-checking Berger code checkers. Integration, the VLSI Journal, 1999, 28: 101

[18] Lala P K. Fault tolerant and fault testable hardware design. Prentice-Hall International, 1985

[19] Lala P K. Self-checking and fault-tolerant digital design. Morgan Kaufmann, 2001