RESEARCH ARTICLE OPEN ACCESS

Design and implementation of error protection scheme for register files using VLSI

S.DHARANI[1], Y.SURESH[2] , N.YUVARANI[3],R.SUPRIYASREE[4]
1(Assistant professor,dharanisadum@gmail.com)
2(Student,suresh29397@gmail.com)
3(Student,nyuvaraniyadav@gmail.com)
4(Student,ridhya.rajput123@gmail.com)
DEPARTMENT OF ECE,ADITYA COLLEGE OF ENGINEERING,MADANAPALLE.

Abstract: Reducing in feature size, energy consumption will leads to increase in vulnerability of microprocessors against soft errors in terrestrial applications. The register file is one of the essential architecture where soft errors can lead to problematic executions. Because, errors may spread rapidly throughout the whole system .Thus, when it comes to reliability register file is one of the major concerns. This paper introduces self immunity technique to improve the integrity with respect to soft errors. If the errors are still found then error correction code is introduced, this process ensures that any error effecting a single bit will corrupt its output for multiple cycles .on the other hand a single bit error in the original copy will corrupt its output for one cycle. We show that our technique can reduce the vulnerability of register file on proposing a soft error mitigation technique for register files.

Keywords:Self Immunity,ECC,Soft Errors,Vulnerability

I. INTRODUCTION

In the early days of computers, "glitches" were an accepted way of life. Since then, as computers have become more reliable (and more relied upon), glitches are no longer acceptable – yet they still occur. One of the most intractable sources of glitches has been the transient "bit-flip", or soft memory error: a random event that corrupts the value stored in a memory cell without damaging the cell itself. Soft errors in electronic memory were first traced to alpha particle emissions from chip packaging materials. Tests and standards have been developed to measure and improve the resistance of memory chips to alpha particles but soft errors have not disappeared.

Register file (RF) is extremely Vulnerable to soft errors Therefore; some processors protect their registers with ECC.

ECC: It is used to verify data transmissions by locating and correcting transmission errors. It is commonly used by RAM chips that include forward error correction (FEC), which ensures all the data being sent to and from the RAM is transmitted correctly.

Self-Immunity: it is to improve the resiliency of register files to soft errors, especially desirable for processors that demand high register file integrity under stringent constraint.

II. LITERATURE SURVEY

The Background and terminology of soft errors proposed several techniques such as mean time between failures, error in time, triple modular redundancy and silent data corruption. these techniques are often used for removing errors in registers and also have vulnerability factors such as architectural and register vulnerability factors

Silent data corruption: Data corruption refers to errors in computer data that occur during writing, reading, storage, transmission, or processing, which introduce unintended changes to the original data. Computer storage and transmission systems use a number of measures to provide data integrity, or lack of errors.

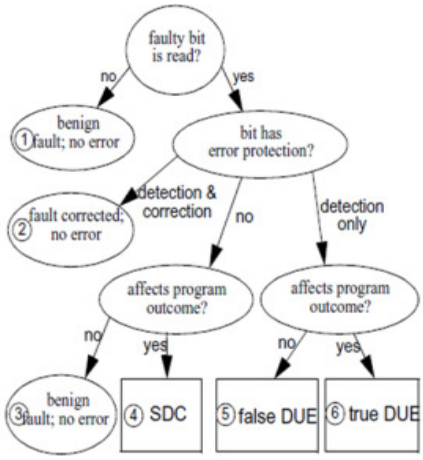

fig:2.1 algorithm of SDC

Vulnerability Factors:

The effective EIT rate per bit is influenced by several vulnerability factors. In general,a vulnerability factor indicates the probability that an internal fault in a device's operation will result in an externally visible error.

The RVF of a register is defined as the sum of the lengths of all its vulnerable intervals divided by the sum of the lengths of all its lifetimes.

$$Vulnerability(reg) = \frac{\sum \text{Vulnerable Interval Time}}{\sum \text{Life Time}}$$

III.PROPOSED TECHNIQUE:

To protect the register file we propose shield and the self-immunity technique that is..,

Shield: Protecting the Register File

Shield performs three operations: to protect the useful lifetime of a register version:

(i) when the register is written, Shield generates and saves the ECCcorresponding to the written data,

(ii) when the register is read, Shield checks whether the register contents are still valid, and

(iii) Shield keeps the protection for the register until it is read for the last Figure shows the architecture of shield.

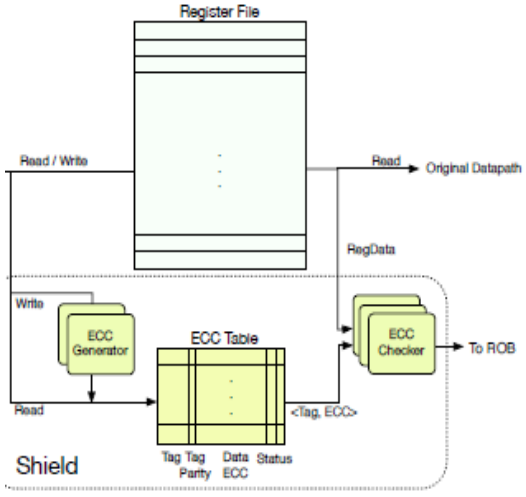

fig:3.1 Architecture of Shield

When an entry is evicted from Shield, its associated register version will no longer be protect

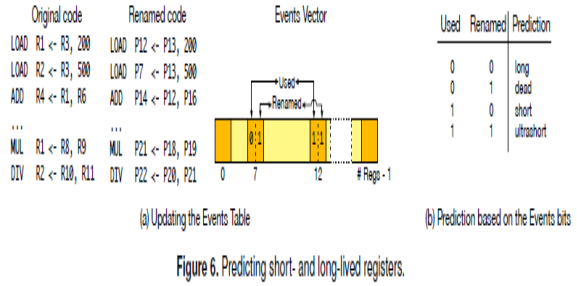

Figure 6. Predicting short- and long-lived registers.

Fig: 3.2 Predicting short and long lived registers

Self-Immunity technique:

It proposes to exploit the register values that do not require all of the bits of a register to represent a certain value. Then, the upper unused bits of a register can be exploited to increase the register's immunity by storing the corresponding SEC Hamming Code without the need for extra bits. The Hamming Code is defined by k, the number of bits in the original word and p, the required number of parity bits (approximately $\log_2 k$). Thus, the codeword will be $(k + \log_2 k + 1)$. Consequently, the condition should be valid $(k + \log_2 k + 1 \leq w)$.Thus, the optimal value of k is 57 in 64-bit architectures and 120 in 128-bit architectures.

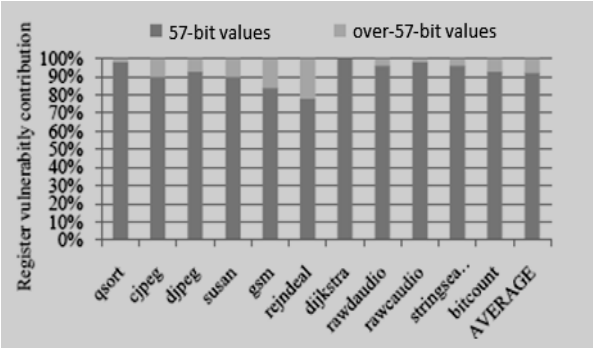

Fig:3.3"57-bit" register values and "over-57-bit" register values in different benchmarks

In addition to the previous key observation, the contribution of "57-bit" register values in the total vulnerable intervals shown in figure

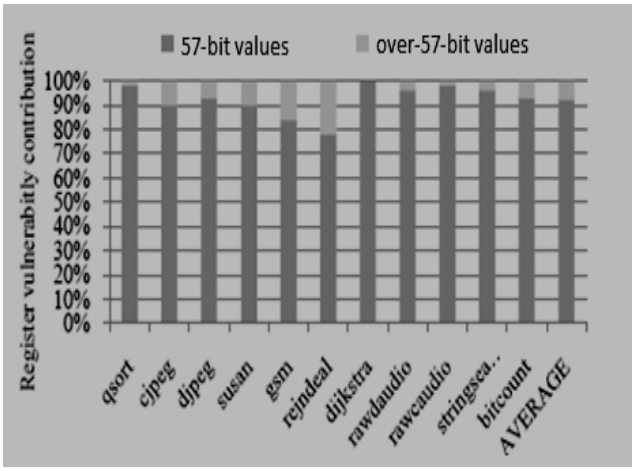

Fig: 3.4 the fraction of vulnerable factor

Architecture for This Technique: The key challenge in distinguishing whether the ECC bits are embedded in the register value or not, is that the processor does not have sufficient information to make this decision when reading a value from a register. Consequently, it needs to distinguish "57-bit" register values from "over-57-bit" register values. To do that, a self-π bit is associated with each register and it initially clears all self-π bits to indicate the absence of any Self-Immunity. For the sake of simplicity, it explains the proposed architecture with the required algorithms in two different steps. writing, reading and correcting data.

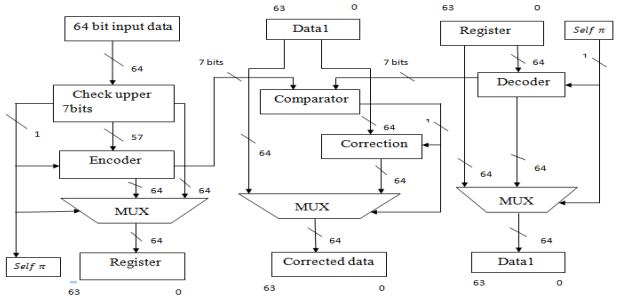

Fig:3.5Micro-architectural support for writing, reading into a register and correcting data

1V.SOFTWARE REQUIREMENTS:

Tool Structure and Flow:

The diagram below illustrates the structure of the ISim tool, and the flow of that tool as it is used to verify a design

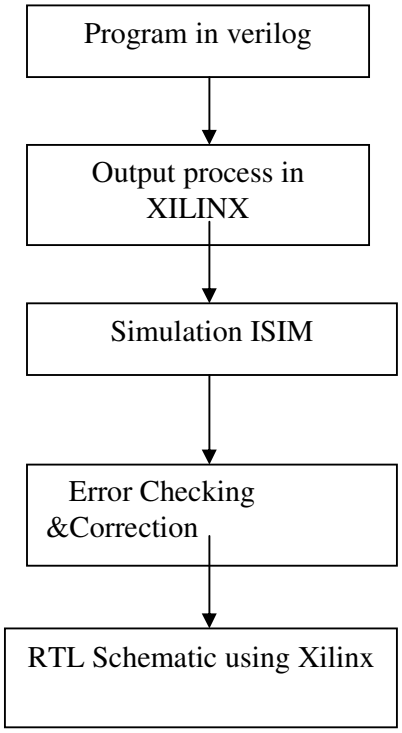

Fig: 4.1Structure of the ISIM tool

Xilinx ISE 13.4:

The Xilinx ISE tools allow you to use schematics**,** hardware description languages *(HDLs)*, and specially designed modules in a number of ways. Schematics are drawn by using symbols for components and lines for wires. Xilinx Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD).

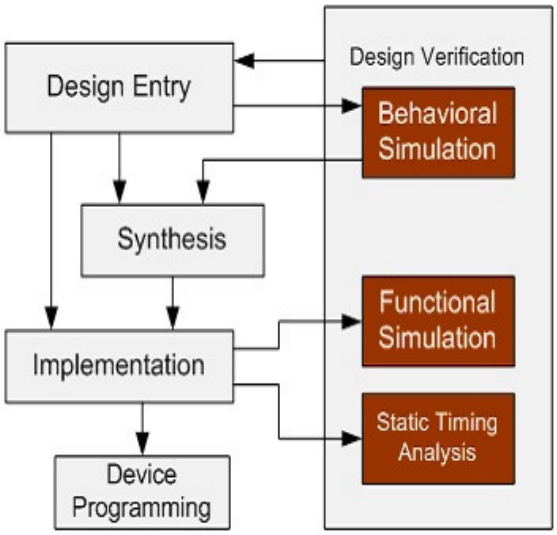

Fig:4.2 ISE Design flow

VHDL design specifications and the steps of this design procedure are listed below:

1. Create Verilog design input file(s) using template driven editor.
2. Compile and implement the Verilog design file(s).
3. Create the test-vectors and simulate the design without using a PLD (FPGA).
4. Assign input/output pins to implement the design on a target device.
5. Download bit stream to an FPGA or CPLD device.
6. Test design on FPGA/CPLD device

A Verilog input file in the Xilinx software environment consists of the following segments:

• **Header:** module name, list of input and output ports.

• **Declarations:** input and output ports, registers and wires.

• **Logic Descriptions:** equations, state machines and logic functions.

•**End:**end

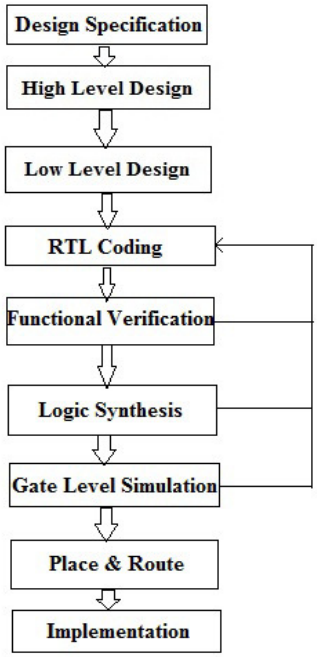

Fig :4.3 Top down design approach

The desired design-style of all designers is the top-down design, shown in figure 4.3 A real top-down design allows early testing, easy change of different technologies, a structured system design and offers many other advantages. But it is very difficult to follow a pure top-down design. Due to this fact most designs are mix of both the methods, implementing some key elements of both design styles.

Verilog supports a design at many different levels of abstractions among those the very important levels are Behavioural level, Register Transfer level, gate level.

V.PERFORMANCE ANALYSIS:

We have coded the self immunity technique in Verilog HDL using the proposed self immunity technique design for bit-width 32. All the de-signs are synthesized in the Xilinx Synthesis Tool and Simulated using Xilinx ISE simulator, the area, delay and power values are compared with conventional self immunity technique.

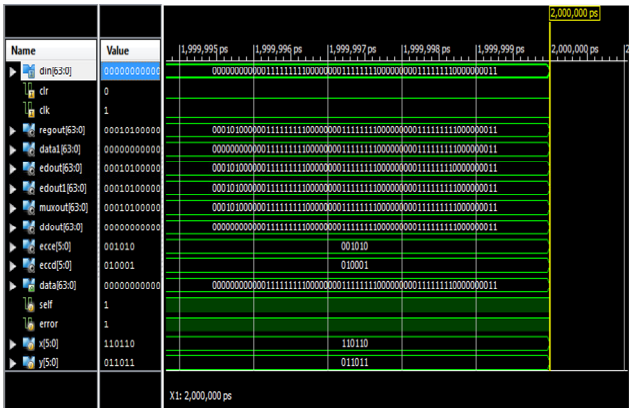

fig:4.4 Simulation results of ECC

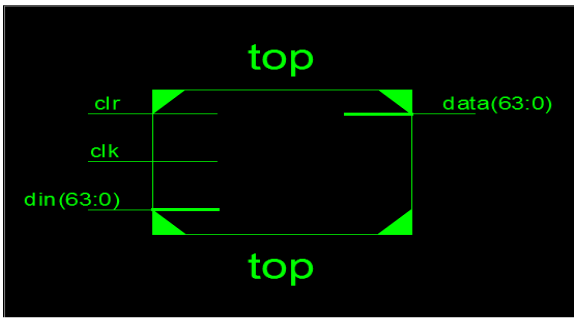

Fig:4.5 RTL with basic inputs and outputs

The synthesis result confirms that the proposed self immunity technique involves significantly high performance and accurate than the existing designs.

Fig: 4.4 Synthesis design

top Project Status			
Project File:	softerror.xise	Parser Errors:	No Errors
Module Name:	top	Implementation State:	New
Target Device:	xc3s500e-4fg320	• Errors:	
Product Version:	ISE 13.4	• Warnings:	
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	150	1920	7%
Number of Slice Flip Flops	64	3840	1%
Number of 4 input LUTs	270	3840	7%
Number of bonded IOBs	130	97	134%
Number of GCLKs	2	8	25%

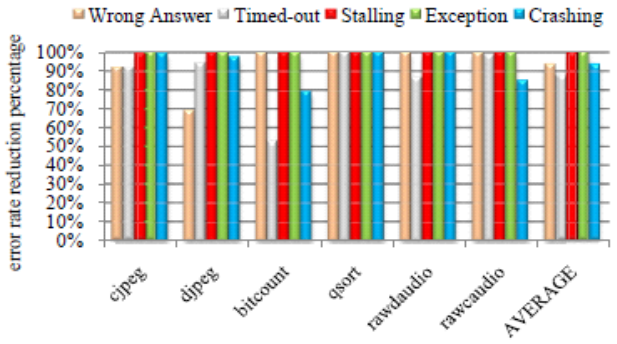

Figure:4.5 Percentage of error rate reduction after implementing this proposed technique.

V CONCLUSION

For embedded systems under stringent cost constraints, where area, performance, power and reliability cannot be simply compromised, we propose a soft error mitigation technique for register files. We developed our project by using 64-bits register file to improve register against soft error and reduce the hacking technique Our experiments on different embedded system applications demonstrate that our proposed Self-Immunity technique reduces the register file vulnerability effectively and achieves high system fault coverage.. It is very highly secure for both transmit & receive data in communication system Moreover; our technique is generic as it can be implemented into diverse architectures with minimum

REFERENCES:

[1]GregBronevetsky and Bronis R.de Supinski,"Soft Error Vulnerability of Iterative Linear Algebra Methods".

[2] RiazNaseer, RashedZafarBhatti, and Jeff Draper, "Analysis of Soft Error Mitigation Techniques for Register Files in IBM Cu-08 90nmTechnology".

[3] M. Kandahar, W. Zhang, and L. Yang, "An area-efficient approach to improving register file reliability against transient errors," in Advanced Information Networking and Applications.

[4] S. Kim and A.K. Somani, "An adaptive write error detection technique in on-chip caches of multi-level cache systems".

[5] K. Walther, C. Galke and H.T. VIERHAUS, "On-Line Techniques for Error Detection and Correction in Processor Registers with Cross-Parity Check".