Fault Tolerant Microarchitecture for an In-Order RISC-V Processor

A Project Report

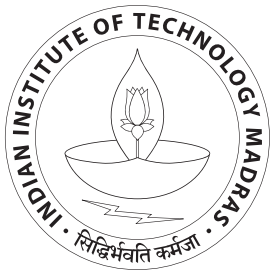
submitted by

SUKRAT GUPTA (EE13M082)

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

under the guidance of
Dr. V. Kamakoti



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2015

THESIS CERTIFICATE

This is to certify that the thesis entitled **Fault Tolerant Microarchitecture for an In-Order RISC-V Processor**, submitted by **Sukrat Gupta (EE13M082)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr.V.Kamakoti
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my guide, **Prof. V. Kamakoti**, CSE department for his guidance, ideas and suggestions. His recommended papers in the field of fault tolerance helped me a lot to come up with an idea for this project work. Thank you for guiding me in this area. Additionally, I would like to thank **Mr. G. S. Madhusudan** for giving me the interesting topic to work on. He provided me enough literature and assisted me whenever needed. I would like to acknowledge **Prof. Nitin Chandrachoodan**, EE department, for serving as my Co-guide.

During my one year at RISE laboratory, I had an opportunity to work with wonderful people. This project work would not be possible without their support. Special thanks to Rajendran M and Neel Gala.

My heartiest thanks to my parents and my wife for supporting me in pursuing my studies here at IITM, this thesis would not have been possible without their understanding and encouragement.

Finally, I am grateful to my organisation VSSC, ISRO for providing me an opportunity to pursue my M.Tech at IITM as sponsored candidate.

ABSTRACT

Deeply scaled CMOS circuits are vulnerable to soft and hard errors. These errors pose reliability concern especially for systems used in space or in harsh environment. Some level of fault mitigation techniques are now the essential requirement for the system design.

This thesis aims to develop a fault tolerant microarchitecture for an in-order RISC-V processor that provides a solution to the reliability issues considering power and area budget. Fault tolerance is provided for the memories, specific registers and ALU which constitute the major part of chip area. Instruction memory, data memory, architectural register file and program counter register are supported with parity bits for Error Correcting Code (ECC). In our approach, data flows along with parity bits throughout the pipeline stages and shall be corrected wherever it is required. The fault handling implementation of ALU is the combination of space redundancy and time redundancy. The fault tolerant ALU comprises 5 types of functional units (FUs) namely adder, multiplier, shifter, comparator and logical unit. All FUs are in dual modular redundant (DMR) configuration with their fault handling logic. Time redundancy technique has been used for isolating faulty FU in case of permanent fault, where different recomputation techniques have been used for different types of FUs. We implement this approach on a 5-stage in-order pipelined processor based on RISC-V ISA (i.e. in-house C-Class Shakti processor) as case study to make it resilient against transient and permanent faults.

Keywords: fault tolerant, single event effects, soft and hard errors, redundancy, pipelined microprocessor

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As complementary metal-oxide-semiconductor (CMOS) technology advances to the nanometer scale, the ever-increasing capability of integrating more and more devices and elements on a single die is boon to the semiconductor industry. Meanwhile, the reliability of the integrated circuits (ICs) product is being severely challenged due to the fact that many previous negligible effects are becoming more prominent, causing significant performance and reliability degradations of nanometer scale integrated circuits.

Moores Law which stipulates that transistor density will double every eighteen months, has been the driving force in advancing CMOS technology. As CMOS technology evolves into the nanometer regime, advanced manufacturing technologies and CAD techniques enable billions of transistors to be fabricated on a single chip, and multiple components will be integrated onto a chip to become a System-on-a-Chip (SoC).While transistor scaling results in high architectural performance , high transistor density and low power consumption, it also increases the concern on the reliability of ICs, specially systems working in space or in harsh environment. As the design complexity continues increasing, power consumption, yield and reliability are three prime challenges for designers in deep nanometer technologies.

Power Consumption: Increasing leakages (band-to-band tunnelling and sub-threshold current), Increasing devices count and high clock frequencies in modern designs constantly lead to an increase of static and dynamic power dissipation, power density and heating at unacceptable levels. These concerns affect any design in general, but they are even more critical in portable and embedded applications due to battery duration and heating dissipation constraints. Clearly, reducing power dissipation has become one of the fundamental constraints for present and future designs. A variety of process-level, architectural-level, circuit-level and software-level approaches have been investigated to deal with this problem. At circuit level, an efficient power reduction approach consists at lowering voltage levels as much as possible. However, this action affects adversely the reliability as noise margin reduces that leads to significant soft

errors and delay/timing faults. Dynamic Voltage Scaling (DVS) has emerged as the most efficient scheme to reduce SoC power dissipation.

Yield: As we scale deeper to nanometric domain, permanent faults related to the fabrication process gain importance and affect severely the fabrication yield of complex SoCs. Such faults are already a concern for embedded memories and require built-in-self-repair (BISR) to maintain acceptable yield for SoCs which include large memory arrays. Memories are early indicators of yield and reliability trends. As to the combinational logic, spot defects create various fault types such as stuck-at faults, short circuits and open circuits etc, which often affect correct operation or increase the circuit delays. With nowadays clock operating speeds, even short delay variations will result in delay/timing faults. Thus, delay/timing faults produced by small spot defects are becoming a severe concern. Furthermore, process parameter variations, both intra-die and inter-die variations in channel length, oxide thickness, gate oxide thickness and flat-band voltage are another important concern.

Reliability: With the channel length shrinks to nanometer scale, reliability degradation has become serious design concern. Several factors conjointly contribute to the reliability degradation, such as radiation-induced soft/hard-error effects, device degradation and fabrication-induced parameter variability. Single event effects (SEE) are now the major concern in secure or critical systems. Meeting reliability specifications will be a critical customer requirement especially in space and nuclear industries. Several factors impact the reliability of the ICs design. First of all, as the device dimensions approaches to the size of atoms, the environmental interferences plays a pivotal role than ever. Secondly, as the supply voltage reaches sub-volt range, noise margin of the semiconductor devices has been greatly reduced. Also, the number of devices on a single die is increasing exponentially with the integration capability; they are operating at a much faster rate as the operating frequency reaches multi-GHz range. This result in cross talk, ground bounce, EMI related problems that lead more transient faults in the system.

Design-for-Reliability (DFR) becomes an important practice to achieve a high level of fault tolerance and the reliable design becomes imperative in the current and next technology generations. Many techniques to improve reliability can incur performance, energy, or cost penalties. However, some solutions targeting at a specific failure mechanism could adversely affect other mechanisms. For example, lowering operational voltage can help mitigate power problems but

increases the vulnerability to soft errors. Hardware redundancy techniques like Triple Modular Redundancy (TMR) increases the reliability of the system by fault masking but on the other hand require significant power and area overhead. Therefore there is always some trade-off between reliability and area/power penalty.

In this thesis, we will provide a fault-tolerant solution for mitigating soft and hard errors in the processor design keeping area and power budget in mind. We will discuss that how traditional Error Correcting Codes (ECC) technique is implemented on memories and registers to give better fault tolerance to the overall system architecture. Our main focus will be on protecting the ALU against transient and permanent faults. The organisation of this thesis is as follows: Chapter 2 introduces an overview of the soft and hard errors in nanometer technologies. Chapter 3 explains our proposed fault tolerant approach and its implementation. Chapter 4 covers the verification methodology adopted for fault tolerant design. Chapter 5 provides overview of simulation set up and FPGA implementation .Chapter 6 concludes the thesis and discusses possible improvements as future work.

CHAPTER 2

SOFT AND HARD ERRORS IN NANOMETER TECHNOLOGIES

With the continuous downscaling of CMOS technologies, the device reliability has become a major bottleneck. The harsh environment of space or nuclear plants can introduce a variety of faults in the logic circuits. The radiations and high energetic particles poses causes soft and hard errors in the system and becoming more prominent now a days. This radiation causes two types of failures namely Total Ionising Dose (TID) Effects and single event effects (SEE) which posing design and test challenges. TID is basically the cumulative long term ionising damage due to electrons and protons deposited in a device. It is measured in Radiation Absorbed Dose (Rads). The device that collects a charged particle for a long time may leads to functional failure. SEE is basically, any measurable effect on the circuit due to an ion strike and these are instantaneous events. SEE basically categorised into soft and hard errors. Both TID and SEE are studied first since they make anomalies in satellite, avionics and nuclear equipments. These effects become one of the most challenging issues that impact the reliability of modern electronic systems used for space applications or even at ground-level applications. This chapter presents an overview of the soft and hard errors in nanometer technologies. The overview of mitigation techniques against soft and hard errors is also presented here.

2.1 Soft Errors

2.1.1 Soft Errors Overview

Radiation-induced soft errors are an increasingly important threat to the reliability of integrated circuits processed in advanced CMOS technologies. Soft error is any measurable or observable change in the state or performance of a microelectronic device, component, subsystem, or

system resulting from energetic particle strike. Soft errors are non permanent, random, non-recurring in nature. When a particle strikes on the transistor then it displaces electrons and holes, thus ionizing a part of the silicon substrate. The displaced electrons and holes begin to recombine and creates a current pulse. The current pulse propagates to other parts of the circuit. When the displaced charge, Qcoll , is more than Qcrit, the pulse is large enough to create a change in state or upset. Qcoll is a function of the ionizing particle's energy, trajectory, point of impact, and the local electric field. The current transient lasts for around 200 picoseconds. Most of the impact is within 2-3 microns of the impact site.
If the effect of soft errors is manifesting up to the system level, it is generally in the form of a sudden malfunctioning of the electronic equipment. Soft errors can not be traced, once memory is updated with new data where corrupted bits were stored earlier. Therefore, failure analysis is not able to conclude that soft errors as the root cause of the problem. Furthermore, the problem is not reproducible or recreated, due to its stochastic or random nature. Therefore,it is usually very difficult to show that soft errors are causing the observed failures.

2.1.2 Soft Errors Classification

Single-event effects (SEEs) are associated with the change of states or transients in a device that energetic external radiation particles induce. Normally, SEEs can be classified into soft and hard errors. Soft errors are non destructive, because resetting or rewriting the device restores normal behaviour thereafter; hard errors are permanent. Soft errors are a subset of single-event effects and can be classified into the following categories:

1. **Single-bit upset (SBU)**: It is also called Single Event Upset (SEU). The event causes a bit flip in a memory cell or a register due to the particle strike. This event causes temporal loss of device functionality and shall be recovered by rewritten the memory or register.
2. **Multiple-bit upset (MBU)**: The event causes the upset of two or more bits in the same word of memory or registers.
3. **Single-event transient (SET)**: The event causes a voltage glitch in the net or wire of the circuit, which propagates and becomes a bit error when captured by a storage element.
4. **Multiple-event transient (MET)**: The event causes multiple voltage glitches in the nets or wires of the circuit, which propagate and becomes multiple bit errors when captured by storage elements.

5. **Single-event functional interrupt (SEFI)**: The event causes the lock-up, reset, or other detectable malfunctioning of a component.

2.1.3 Soft Error Specification

Generally, Soft Error Rate (SER) is the rate at which a device or system encounters or is predicted to encounter soft errors. SER is measured in units of Failures In Time (FIT), where 1 FIT denotes one failure per billion device hours (i.e. one failure per 114,115 years). The additive property of FIT makes it convenient for calculations, but mean time to failure (MTTF) is often more intuitive. MTTF is inversely related to FIT and it is not additive. A FIT rate of 1000 is equivalent to MTTF of 114 years. A single bit has fault rate of about 1-10 mFIT then 1 GB memory has fault rate of 109*8*0.01 = 8*107 FIT i.e. an error every 12.5 hours.
Researchers expect about an 8 percent increase in soft-error rate per logic state bit each technology generation. Since the number of logic state bits on a chip is following Moores law, and the aggregate effect on soft-error FIT on a chip is shown in Figure 2.1. Notice that by the 16nm generation, the failure rate will be almost 100 times that at 180nm technology. In an electronic component, the failure rate induced by soft errors can be relatively high compared to other reliability issues.

2.2 Hard Errors

2.2.1 Hard Errors Overview

Hard errors are the events which interrupt device function and permanently damage the device. They are destructive in nature and cause permanent damage to the device. Under hard error when a bit can never be read or written reliably, the processor might not be able to correct and recover from the error. Hard errors may be caused by wire wear out due to electro-migration, gate oxide wear-out and ageing defects that increase over time. This cause a open or short mode failure.

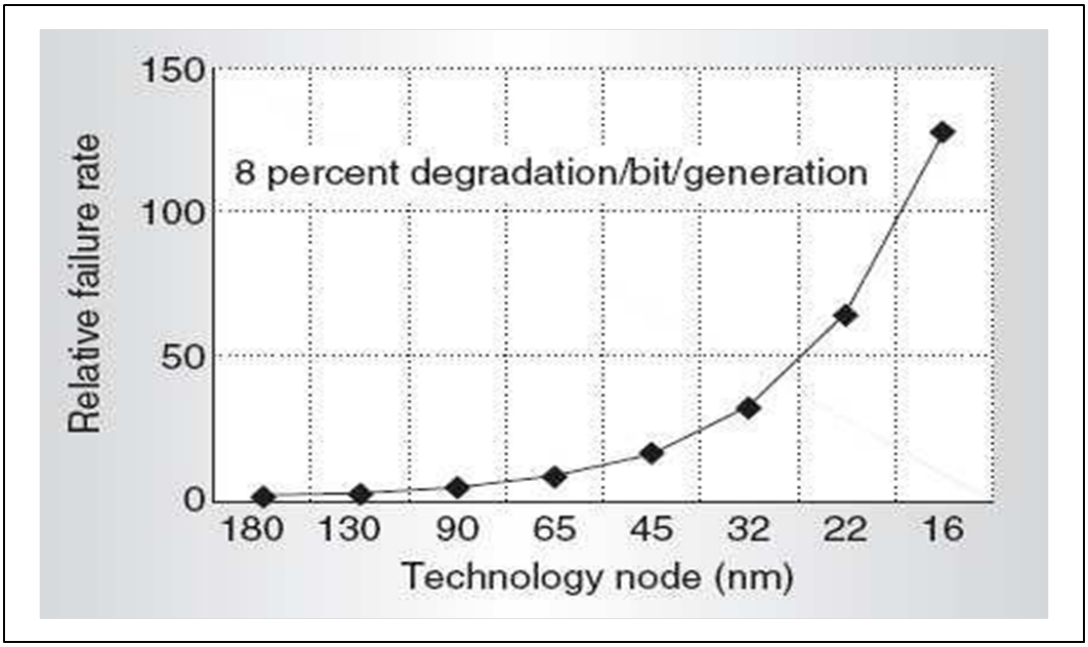


Figure 2.1: Soft Error failure-in-time of a chip (Figure taken from [8])

2.2.2 Hard Errors Classification

Hard errors are a subset of single-event effects and can be classified into the following categories.

1. **Single Event Latch Up (SEL)**: An abnormal flow of high-current in a device caused by the passage of a single energetic particle through sensitive regions of the device structure and resulting in the loss of device functionality. It is triggered by heavy ions, protons, neutrons and cause catastrophic failure.
2. **Single Event Gate Rupture (SEGR)**: An event in which a single energetic-particle strike results in a breakdown and subsequent conducting path through the gate oxide of a MOSFET. An SEGR is caused by an increase in gate leakage current which finally cause either the degradation or the complete failure of the device.
3. **Single Event Burn out (SEB)**: An event causes direct path between source and drain of the transistor. This cause a local hotspot and finally failure of the device.

2.3 SEE in Integrated Circuits

Digital systems are typically divided into two subsystems - combinational subsystems and sequential subsystems (composed of memory elements). Processor is basically a Finite State

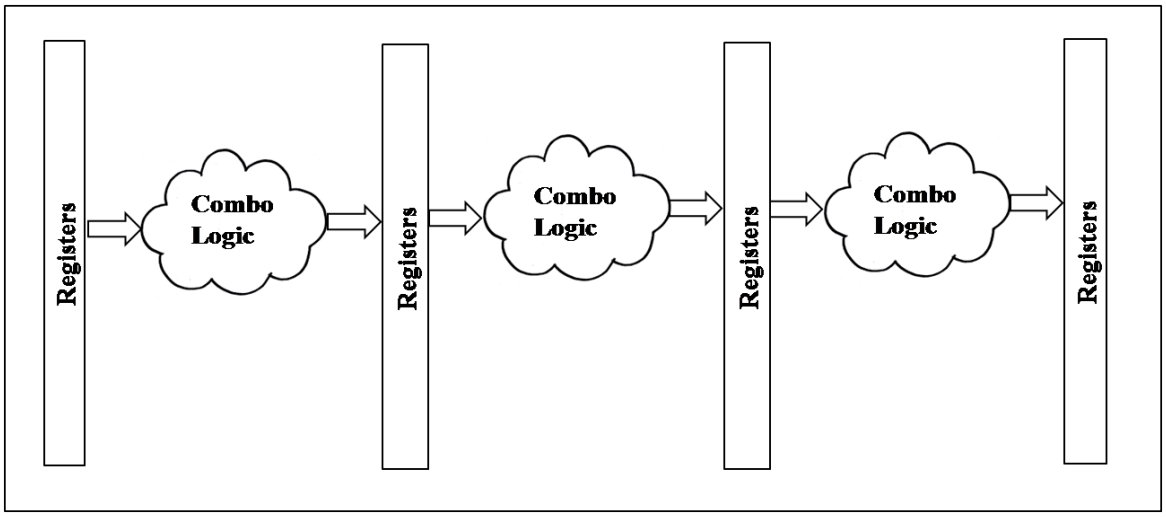


Figure 2.2: FSM Representation of the Processor

Machine (FSM) and shall be represented as shown in Figure 2.2. Combinational logic and memory elements are affected by ionizing radiation quite differently; however both contribute to the SEE rate of digital systems.

SEEs induced by memory elements can occur when radiation particles strike memory elements directly. However, SEEs induced by combinational elements can occur only if the SETs generated when radiation particles strike a vulnerable combinational node, are latched by a memory element or energetic particle causes permanent open or short mode failure.

2.3.1 SEE in Memory Elements

Memory elements comprises SRAMs, DRAMs, Latches, Flip Flops etc. SRAMs are non volatile memory therefore their storage signals do not need to be refreshed periodically. SRAMs use a bistable latching circuitry to store bit information. Similarly, Flip Flops, registers are basically back to back inverter which stores a bit. A particle strike on any of the nodes may cause the node to transition. If this transient glitch propagates through the inverters, it causes the wrong value to be latched. When this happens, external circuitry is needed to rewrite the nodal value. Latches also contain a regenerative path similar to SRAMs, thus latch-induced SEUs are largely based on the same principles as SRAM-induced SEUs. Flip-flops typically consist of two latches in a master-slave setup thus radiation particle strikes on FFs have similar effects as

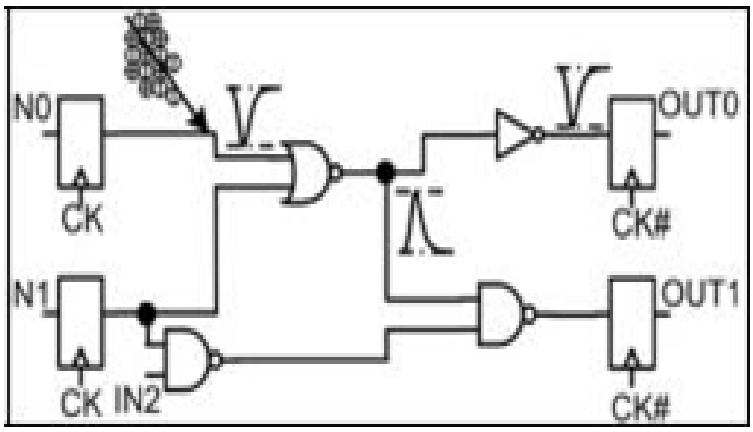


Figure 2.3: SEU propagating through a logical circuit

in latches and SRAMs.

2.3.2 SEE in Combinational Logic

As in memory elements, charge collection occurs at a combinational node affected by a radiation particle strike. If the collected charge exceeds the Qcrit, a 100 to 200 picosecond (ps) wide transient voltage glitch is generated. The transient voltage may propagate through combinational elements and be latched by a memory element as seen in Figure 2.3. However, three masking effects present in digital circuits generally prevent transients from becoming SEUs.

1. **Logical Masking**: The logical masking effect can be described with the NAND gate seen in Figure 2.4(a). If a particle strikes one of the input nodes of the NAND gate, but the other input remains in the controlling state (0 in this case), the output of the gate will not change and the SEU will be completely masked. For an SEU to propagate through a combinational logic element, sensitive path must exist from the affected node to the output of the logic element.

2. **Electrical Masking**: All CMOS circuits have limited bandwidths. SEUs with bandwidths higher than the cut-off frequency of a circuit will be attenuated. This causes the amplitude of the SEU to reduce, the rise and fall times to increase, and, eventually, the pulse to disappear as it propagates through logic elements as seen in Figure 2.4(b).

3. **Latching Window Masking**: This effect, also known as Temporal Masking, can be described with Figure 2.4(c). As the SEU moves towards the D input node of the FF, it might show up outside the latching window of the FF preventing it from being latched, thereby preventing an SEU from occurring. In FFs and latches, setup and hold time requirements make up the latching window. As the operating frequency for subsequent technology nodes increases, the latching window decreases due to a resulting decrease in

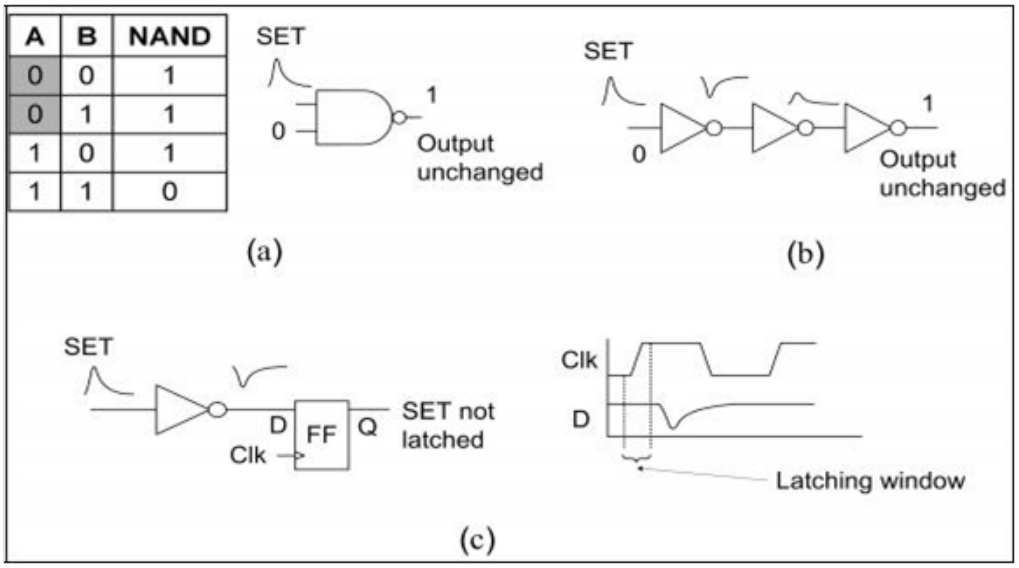


Figure 2.4: Different SEU masking effects in combinational logic

setup and hold time requirements, thus the effect of latching window masking decreases, effectively causing an increase the SEU rate.

2.4 Contribution of Memory Elements and Combinational Logic to SEE Rate

We have determined the mechanisms through which combinational logic and memory elements generate Single Event Upsets(SEUs), however it is important to understand the relative contribution of combinational and sequential elements to be able to choose a suitable protection technique against SEEs. Conventionally,the SEE rate of CMOS designs is largely dominated by sequential elements for low frequency circuits primarily because the three masking effects observed in combinational logic often prevent SETs from being latched by a memory element. Another reason for the domination of sequential element SEUs is that memory elements usually have a lower Qcrit than combinational elements, thus a radiation particle is more likely to cause an SEU in a sequential circuit than in a combinational circuit. However,the distinction between the relative contribution of combinational logic and memory elements to SEE rate is not always as clear-cut as implied. The combinational logic area is typically greater than the

sequential area; this makes combinational elements more probable to be effected by radiation particles. Therefore, it is possible that increasing the combinational logic area exponentially can cause combinational elements to dominate SEE rate.

Research studies have shown that the SEE rate of ICs is frequency dependent. Radiation effects experts postulate that as the operating frequency of CMOS devices continues to increase, an increase in SEE rate is observed. For low frequency applications, sequential logic errors dominate combinational logic errors. However, as sequential logic error rate is relatively independent of frequency, combinational logic error rate dominates for high frequency ICs, increasing the overall error rate. Much of this increase in combinational logic induced SEUs can be attributed to a weakening of the latching window masking effect at high frequencies. Since set-up and hold time, which typically makes up the latching window, has to be less than clock frequency, latching window decreases with an increase in clock frequency. Another reason for combinational logic error domination at high frequencies is that as transistors switch faster, the effects of electrical masking diminishes.

We may face the next barrier i.e. hard errors. Hard errors in combinational logic cannot be neglected due to stress and hostile environment system may be exposed to. Wear out, device degradation, avalanche effects etc. poses a probability of permanent failure in deep nanometer technologies. We cannot neglect permanent failure in combinational logic in critical systems which may cause catastrophic failure.

2.5 Fault Mitigation Techniques

Various techniques at different levels exist to mitigate the TID and SEE effects on the system. These methods are broadly categorised into two types as mentioned below.

2.5.1 Fault Avoidance

Fault avoidance is refer to the mechanics which is preventing the system from the occurrences of faults, in other words, in-built fault tolerance exists in the system. The main techniques for fault tolerance are as follows :

1. **Shielding** : This involves the protective coating, so that energetic particles cannot penetrate. This can be done at chip level by selecting suitable chip packaging material or at system level also.

2. **High Quality Fabrication Material**: The high quality material and ultra clean fabrication facility make the device contaminant free and avoid the faults which may occur later.

3. **Radiation Hardened Process Technologies**: This refers to avoid the faults by process selection. Specific methods such as use additional well isolation, replace bulk silicon with SOI are commonly used to make radiation hardened design.

2.5.2 Fault Tolerance

1. **System Level Tolerance**: Various methods exists at system level to compute reliably in harsh environment. The use of redundancy, self checking hardware, watchdog timers are the most common methods found at both hardware and software level. Table 2.1 summarizes the common fault tolerance methods.

2. **Error Detection and Correction Logic (EDAC)**: Sometimes, it is impractical to use above mentioned mitigation techniques to avoid SEEs. EDAC techniques like cyclic redundancy check (CRC), hamming code, convolution encoding are mostly preferred in memories where parity bits are added to each memory word. Depending upon the level of fault tolerance, number of parity bits increases. Mostly Single Bit Error Correction and Double Bit Error Detection (SEC-DED) based on hamming codes is commonly followed to achieve fault tolerance in memory based systems.

Protection Methods	**Capability**
Watchdog Timer	If a refresh pulse is not generated within a specified period means faulty condition. Perform the required action.
Dual Modular Redundancy (DMR)	Two equivalent systems working on same data, if the two system disagrees, then system reset shall be performed.
Triple Modular Redundancy (TMR)	Use 3 systems to perform the same operation. Voter logic to decide the outcome and mask the fault.
Time Redundancy	A system is redoing the same operation more than once to mitigate the transient fault.
Lockstep	Two devices in the system are clock synchronised and working on same input. If devices disagrees, then system reset shall be performed.

Table 2.1: Summary of Fault Tolerance Methods

CHAPTER 3

PROPOSED FAULT TOLERANT MICROARCHITECTURE

Due to the fact that soft errors are non-persistent, they will not be able to be detected when the environmental noise source disappears. This makes the detection, diagnosis and correction of soft error effects relatively difficult. With the technology shrinking at faster rate, hard errors cannot be overlooked. Ever since the discovery of the soft and hard errors, researchers in universities and semiconductor companies have spent tremendous efforts and resources on the techniques and methodologies to prevent it from causing damage to electronic product. We have studied some of previous work done in fault tolerance area. In this chapter, we proposed our idea to achieve fault tolerance in the in-order RISC-V processor design and showing our main contributions.

3.1 Soft Error Mitigation Techniques

They can be roughly classified into three distinct categories as explained below.

3.1.1 Technology and device-level

This technique requires fundamental changes to the underlying fabrication technology used to manufacture ICs. This is also called Radiation Hardening by Process (RHBP) where process such as Silicon on Insulator (SOI) or Silicon on Sapphire (SOS) etc has been used. This solution is viable at the expense of additional process complexity, slow process, yield loss and substrate cost.

3.1.2 Circuit-level

This technique relies on changes in the circuit design to reduce soft error sensitivity. It involves designing the circuit in such a way that it can combat the radiation effects. This is achieved by adding redundancy at transistor level or circuit level.

3.1.3 Architectural and System-level

This technique deal with soft errors at the system architecture level. These solutions may be more effective than circuit level techniques as the definition of what constitutes an error typically lies in the architecture (e.g., a strike on a Inter Stage Buffer does not result in an error in a microprocessor). The use of error detection and correction (EDAC) is by far the most effective method of dealing with soft errors in memory component. Hence, the soft-error mitigation techniques based on architectural and system level are mainly applied on processor-based systems.
In the following section, we will illustrate a detailed overview of proposed schemes at architectural and system level aiming to mitigate soft and hard errors in a processor-based systems.

3.2 Proposed Fault Mitigation Techniques for 32 bit In Order RISC-V Processor

The processor-based electronic systems include two basic parts: memory arrays and combinational logic. Due to the fact that soft error rate may exceed the FIT specifications in various application domains, we have come up with the mitigation techniques that hence, in such applications, soft-error mitigation schemes should be employed for memories and eventually for combinational logic.

1. Single bit error correction and double bit error detection (SECDED) has been provided for Instruction Memory (IMEM), Data Memory (DMEM), Architectural Register File (ARF) & Program Counter register (PC).
2. Instruction, Data, Registerfile data & program counter flows through the pipeline stages

alongwith their checkbits, therefore SEU at Inter Stage buffers (ISB) also been taken care.

3. The most important component of processor i.e. the execution unit is made completely fault tolerant against transient error and permanent faults. The execution unit comprises the ALU in Dual Redundant configuration with fault detection and isolation (FDI) of faulty unit using recomputing methods (discussed in detail later).

3.3 Soft and Hard Error Mitigation Techniques for Memories and Registers

Memories represent the largest parts of modern designs. In addition, they are more sensitive to ionizing particles than logic, since they are designed to reach the highest possible density. As a matter of fact, the SER of modern designs is dominated by the SER of their memories. To reduce the SER of a processor design, protecting memories is the first priority, therefore in our design instruction memory, data memory, registerfile, inter stage buffer & program counter register are considered for EDAC.

The simplest solution for error detection is to add parity bit (even or odd) for each data word. Whenever data are retrieved, a check is run comparing the parity of the stored data to its parity bit. If a single error has occurred, the check will reveal that the parity of the data does not match the parity bit. Thus, the parity system allows for the detection of a soft error for a minimal cost in terms of circuit complexity and memory width (only a single bit is added to each word). But here the disadvantage is that it can only detect and cannot correct the error. For processor based system, we need correction so that processor keeps on executing the instruction in case of SEE therefore EDAC or ECC is employed in our design.

In general, the single error correction double error detection (SEC-DED) schemes are considered sufficient as most of the soft errors events are single bit errors. Error correction is achieved by adding extra bits to each data vector encoding the data so that the *hamming distance* between any two possible data vectors is, at least, three for Single Error Correction (SEC) i.e $d_{min} = 3$. The minimum hamming distance required for SEC-DED is 4 (i.e. $d_{min} = 4$).The general equation followed for k-parity bits required for n-bit data words for SEC is $n + k \leq 2^k - 1$. Therefore, for $n = 32, k = 6$ *or* $n = 64, k = 7$.

S.No	No. of Data Words	No. of parity bits for SEC	No. of parity bits for SEC-DED	Area Penalty for SEC-DED
1	8	4	5	62.5%
2	16	5	6	37.5%
3	32	6	7	21.8%
4	64	7	8	12.5%

Table 3.1: Parity Bits for SEC-DED

From the above discussion, we observe that, single error correcting codes are very convenient for modern designs using large data widths, since their cost for large word widths becomes moderate. EDAC or ECC protection will provide a significant reduction in soft failure rates but at a higher cost in terms of design complexity, the additional memory required, and the inherent latency introduced during access, parity generation and correction.

3.3.1 Implementation

In our design, SEC-DED modules are placed in the design after ISB and before the combinational logic at appropriate pipeline stages so that any single bit error before feeding to combinational logic shall be corrected. Similarly after getting result from combinational logic, checkbits have been generated for data and this will flow to the next level of pipeline. In this way, SEE on the pipeline registers or ISB is also covered. The conceptual diagram is shown in Figure 3.1. As discussed above for 32-bit data words, we need 7 parity bits for SEC-DED. In our design, we took 8 parity bits for checkbits for 32-bit dataword keeping last checkbit always 0. This is just done for simplification in understanding as lower 8 bits are checkbits and upper 32 bits are information bits in 40 bit data. The length of each word in instruction memory, data memory, register file and program counter register are 40 bits where upper 32 bits are instruction or data and lower 8 bits are checkbits.

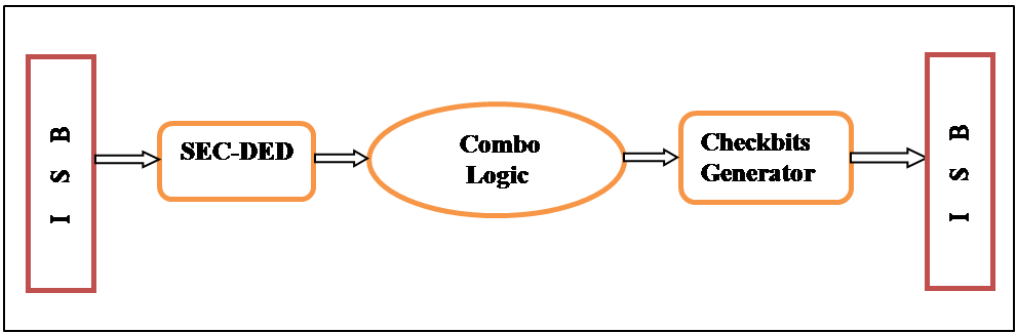


Figure 3.1: Placement of SEC-DED Logic in the pipeline

3.3.2 Error Correcting Code Logic

This module used the reference from the application note provided by Xilinx [10], which describes the implementation of an Error Correction Control (ECC) module in a Virtex-II Pro, VirtexTM-II, Virtex-4, or Virtex-5 devices. The design detects and corrects all single bit errors (in a codeword consisting of 32-bit data and 7 parity bits), and it detects double bit errors in the data (SEC-DED). This design utilizes Hamming code, a simple but powerful method for ECC operations. As a result, this design offers exceptional performance and resource utilization. The participating data bits for the computation of check-bits are as follows:

$$chkbit[0] = d[0] \oplus d[1] \oplus d[3] \oplus d[4] \oplus d[6] \oplus d[8] \oplus d[10] \oplus d[11] \oplus d[13] \oplus d[15] \\ \oplus d[17] \oplus d[19] \oplus d[21] \oplus d[23] \oplus d[25] \oplus d[26] \oplus d[28] \oplus d[30]$$

$$chkbit[1] = d[0] \oplus d[2] \oplus d[3] \oplus d[5] \oplus d[6] \oplus d[9] \oplus d[10] \oplus d[12] \oplus d[13] \oplus d[16] \\ \oplus d[17] \oplus d[20] \oplus d[21] \oplus d[24] \oplus d[25] \oplus d[27] \oplus d[28] \oplus d[31];$$

$$chkbit[2] = d[1] \oplus d[2] \oplus d[3] \oplus d[7] \oplus d[8] \oplus d[9] \oplus d[10] \oplus d[14] \oplus d[15] \oplus d[16] \\ \oplus d[17] \oplus d[22] \oplus d[23] \oplus d[24] \oplus d[25] \oplus d[29] \oplus d[30] \oplus d[31];$$

$$chkbit[3] = d[4] \oplus d[5] \oplus d[6] \oplus d[7] \oplus d[8] \oplus d[9] \oplus d[10] \oplus d[18] \oplus d[19] \oplus d[20] \oplus d[21] \oplus d[22] \oplus d[23] \oplus d[24] \oplus d[25];$$

$$chkbit[4] = d[11] \oplus d[12] \oplus d[13] \oplus d[14] \oplus d[15] \oplus d[16] \oplus d[17] \oplus d[18] \oplus d[19] \oplus d[20] \oplus d[21] \oplus d[22] \oplus d[23] \oplus d[24] \oplus d[25];$$

$$chkbit[5] = d[26] \oplus d[27] \oplus d[28] \oplus d[29] \oplus d[30] \oplus d[31];$$

$$chkbit[6] = d[0] \oplus d[1] \oplus d[2] \oplus d[3] \oplus d[4] \oplus d[5] \oplus d[6] \oplus d[7] \oplus d[8] \oplus d[9] \oplus d[10] \oplus d[11] \oplus d[12] \oplus d[13] \oplus d[14] \oplus d[15] \oplus d[16] \oplus d[17] \oplus d[18] \oplus d[19] \oplus d[20] \oplus d[21] \oplus d[22] \oplus d[23] \oplus d[24] \oplus d[25] \oplus d[26] \oplus d[27] \oplus d[28] \oplus d[29] \oplus d[30] \oplus d[31];$$

$$chkbit[7] = 0;$$

$$Syndrome\ Pattern = Checkbit\_in \oplus Recomputed\ Checkbits$$

Data of IMEM, DMEM, ARF and PC register are stored along-with their check-bits. While reading the data, SEC-DED logic recomputes the check-bits and compared with check-bits bundled with data. Syndrome pattern is generated from the comparison of recomputed check-bits and stored check-bits. By decoding the syndrome pattern, the erroneous bit shall be traced and corrected accordingly. The flag such as singleerr, checkbiterr & doublebiterr are raised on error. If singlebiterr = 1, means there is single bit error happened in the data and has been corrected. If doublebiterr = 1, means there are double bit or multiple bit errors happened and shall be detected only. If checkbiterr = 1, means there is single bit error happened at check-bit positions. While writing the new data in DMEM, ARF & PC register, check-bits are generated and bundled with data for subsequent data flow in the pipeline.

3.3.3 Summary

- Instructions are stored with corresponding check-bits in instruction memory (IMEM). Instructions are read in fetch, decode and execute stage therefore, passed through SEC-DED logic in these stages.
- Data with check-bits are stored in Data Memory (DMEM). Data of DMEM is passed through SEC-DED logic in MEM stage.
- Check-bits are generated for the ALU result (i.e. RAW data), ALU results with check-bits are then used for operand forwarding, storing the data, writing of registerfile etc.
- Program Counter register is also updated with corresponding check-bits and it has been corrected before its usage in fetch stage & execution stage.
- Data flows along with check-bits throughout the pipelines stages. Therefore, soft errors in the ISBs are also tolerated.

3.4 Soft and Hard Error Mitigation Techniques for Execution Unit

Single event upsets (SEU) that are caused by cosmic particle strikes to combinational logic have been studies by several researchers. These upset can occur when the particle strikes at particular node and deposit the charges. This leads to voltage disturbance on that node. If the voltage disturbance happens near the clock edge then the wrong value may be latched, causing the store data to be incorrect. SEU on the combinational nodes do not pose the problem if the clock period is high compared to the duration of SEU (nanoseconds vs picoseconds). As the clock frequency increases the probability of such upsets also increases. At high clock frequency (i.e Ghz range) the contribution of upsets from combinational logic may even exceed that of memory elements. The SEE can cause the permanent failure also where particular node value becomes permanent '0' (short to Gnd) or '1' (short to Vdd). Deep sub micron technology further increases the SEE rates.

The execution unit or the ALU is considered as largest combinational logic in the processor design and typically occupy 40%-50% of chip area. **Our design of execution unit can tolerate transient failures as well as 1 permanent failure per functional unit (FUs are adder, multiplier, shifter, comparator and logical unit)**.

3.4.1 Implementation

There are basically 5 types of FUs namely adder, multiplier, shifter, comparator and logical unit in the fault tolerant ALU (ft_alu) design. Each type of FU is in DMR configuration with fault handling logic called fault tolerant FU as shown in Figure 3.2. The FU-1 and FU-2 are prime and redundant functional units respectively. Faulty FU shall be isolated using time redundancy technique. The proposed scheme can tolerate one FU failure/type i.e. failure of one adder circuitry, one multiplier circuitry, one shifter circuitry, one comparator circuitry and one logical circuitry can be tolerated. The 5 types of fault tolerant FUs in our design are named ft_adder, ft_multiplier, ft_shifter, ft_comparator and ft_logical and can perform the computation in single cycle. It should be noted that separate ft_adder has been used for effective address

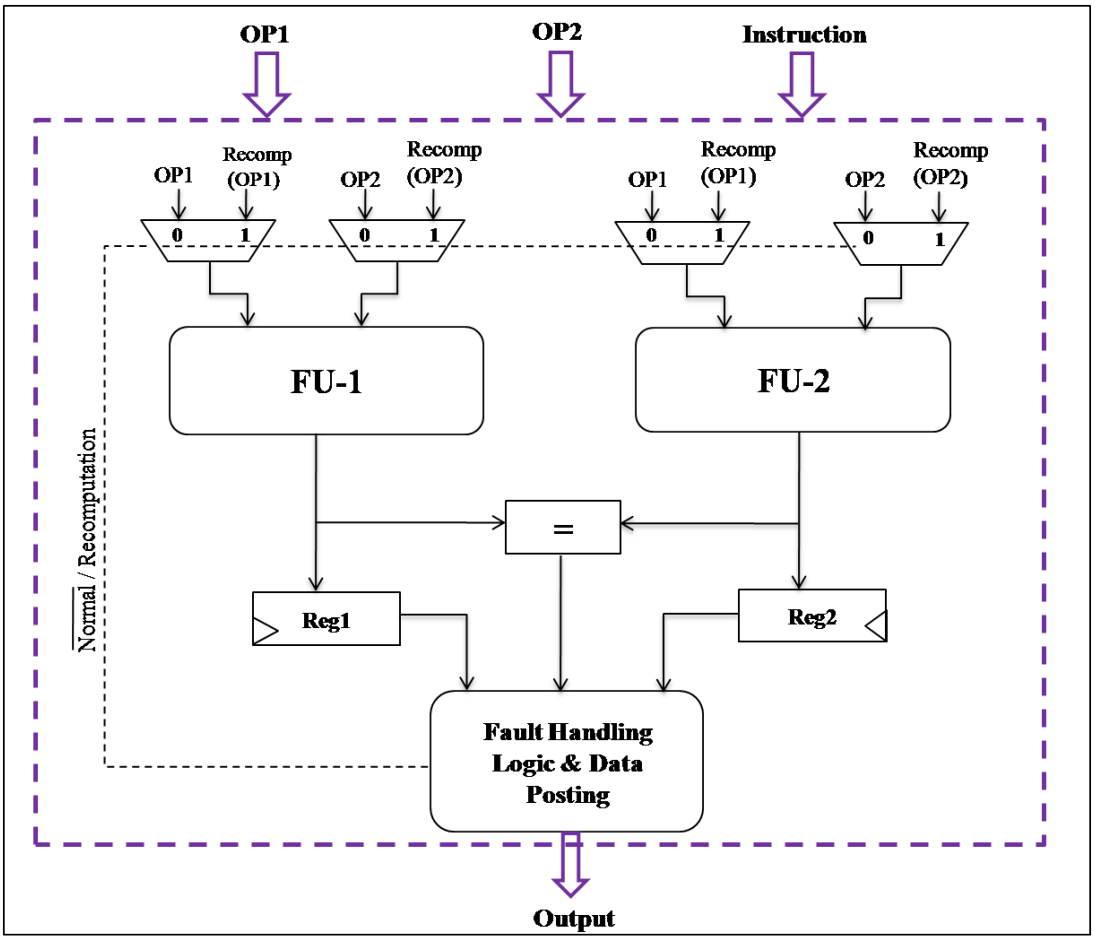


Figure 3.2: Block Diagram of Fault Tolerant FU

calculation in case of branch/jump and memory related instruction. The instructions handled in these fault tolerant FUs of ALU are shown in Table 3.2.

3.4.2 Fault Handling Logic

Definitions and Assumptions:

No error (NE): Match between the outputs of FU-1 and FU-2 is considered as No Error.

Transient Error (TE): Mismatch between the outputs of FU-1 and FU-2 is considered as Transient Error.

Permanent Error (PE): If transient error persists for 5 consecutive clock cycles, then we consider a permanent fault condition. Our assumption is that PE finally leads to bit stuck at fault at the output of faulty FU.

Fault Tolerant FUs	Instructions handled	Remarks
ft_add_sub	LUI,AUIPC,ADDI,ADD,SUB	Addition or Subtraction related instructions
ft_adder	Jump, Branch, Load & Store related instructions	Used only for addition for Effective Address Calculation (EAC)
ft_comparator	SLT,SLTI,SLTU,SLTIU	Comparator related instructions
ft_multiplier	MUL,MULH,MULHSU,MULHU	Multiplication related instructions
ft_shifter	SLLI,SRLI,SRAI,SLL,SRL,SRA	Shift related instructions
ft_logical	XORI,ORI,ANDI,XOR,OR,AND	Bitwise operation related instructions

Table 3.2: Instruction Handling in Fault Tolerant FUs

Any error in the execution will be detected by comparing the outputs of FU-1 and FU-2. If the results of both FU matches then the result will be posted to the next stage of pipeline otherwise it will be considered as transient error. In case of transient error, no data will be posted and will lead to stall in the 5- stage rigid pipeline. If transient error persists for 5 consecutive cycles, we will be considering a permanent failure of one or both FU. To identify the faulty FU, time redundancy technique has been used which involves different recomputing methodology for different FUs in other words recomputation with different coding schemes for addition or subtraction, multiplication, shift operation,comparator related operations and logical operations. In the 6th cycle, Recomputation will be done and faulty FU will be isolated. Here onwards always post from one available FU. The fault handling implementation is describes in Table 3.3 and 3.4

The output of fault tolerant unit is *Data, Destination Register, Health Flag, Valid Bit*.
Data: It is desired result obtained as per the execution of instruction.
Destination Register: The identity of register where the data will be written in writeback stage.
Health Flag bits: The 4-bit Composite health flag information also available along with posted data in Execute Stage which tells about type of error in the FUs executing the current instruction. {Bit3 = 1/0: Permanent/No failure }, {Bit2 = 1/0: Transient/No Error},{Bit1 = 1/0: FU-2 faulty/Not faulty},{Bit0 = 1/0:FU-1 faulty/Not faulty}
Valid Bit: This bit is passed to decide whether to deque the ID_EX-ISB. If valid is false then

Cycle	Equality Check	Error Type	Action Taken
1	Match	NE	Post data from FU-1
2	Mismatch	TE	No posting (Redo the operation)
3	Mismatch	TE	No posting (Redo the operation)
4	Mismatch	TE	No posting (Redo the operation)
5	Mismatch	TE	No posting (Redo the operation)
6	Mismatch	TE	No posting (Redo the operation)
7	Not done as re-computing cycle	PE	Recomputation to identify faulty FU. Update the PE flag to decide data selection and posting. Here onwards post the data according to PE flag table (Table 3.4)

Table 3.3: Fault Handling

PE Flag of FU-1	PE Flag of FU-2	Conclusion/Action Taken
0	0	Not able to detect faulty FU, Post data as 0. Interrupt Service Routine (ISR) to decide next course of action.
0	1	FU-2 faulty. Isolate FU-2 and Post data from FU-1 here onwards.
1	0	FU-1 faulty. Isolate FU-1 and Post data from FU-2 here onwards.
1	1	Both FU-1 and FU-2 are faulty. Post data as 0. Interrupt Service Routine (ISR) to decide next course of action

Table 3.4: PE FLAG

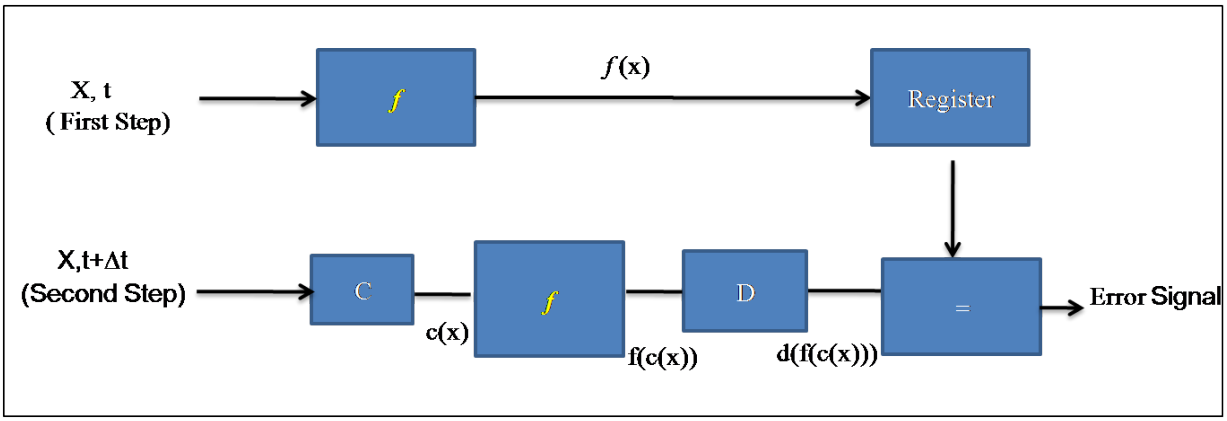


Figure 3.3: Recomputing Methodology (Figure taken from [1])

ISB is not dequeued lead to stall in the pipeline. In case of both units failure or not able to identify failed unit, we are assigning valid bit true so that processor will not stalled permanently in such cases using the health Flag , Interrupt Service Routine (ISR) shall decide action to be performed.

3.4.3 Recomputing Techniques and Fault Detection Capability

In this section, we will discuss about the recomputing method used for different functional units (FUs) and their fault detection capability. Recomputing uses the principle of time redundancy in detecting the errors as shown in Figure 3.3. Here C and D are called coding & decoding functions. If C and D are properly chosen then fault in Unit f will affect $f(x)$ and $f(c(x))$ differently therefore the output of first step (Normal Computation) and second step (Recomputation) will not match leads to error. Different recomputing techniques have been implemented for different FUs in the ALU design as discussed in detail in following sections.

3.4.3.1 Adder Functional Unit

Addition and subtraction related instructions are handled by this FU. As sum and carry are self dual functions, in normal computation, first operand $(OP1)$ and second operand $(OP2)$ are

given to the adder circuitry which computes output as

$$F_n = OP1 \pm OP2 \ (Carry_{in} = 0)$$

In recomputation step, operands in complement form will be given to the adder circuitry i.e. $Recomp(OP1) = \overline{OP1} + 1$ (1 is added here to absorb $Carry_{in} = 1$) and $Recomp(OP2) = \overline{OP2}$. The output of recomputation step will be

$$F_r = (\overline{OP1} + 1) \pm (\overline{OP2})$$

The conceptual block diagram of fault tolerant adder is shown in Figure 3.4.

No Fault Condition: The inputs to adders are OP1 (i.e.First Operand) & OP2 (i.e.Second Operand). As there are no faults in adder1 and adder2 circuitry, the output of both adders will match and data of adder1 is posted. The health flag should be "0000".

Transient Fault Condition: If the outputs of adder1 and adder2 circuitry are not matches in current cycle then error count becomes '1' and fault handling logic will not post the output. This leads to stall in the pipeline. The outputs will again be matched in next clock cycle if it matches then error count will reset to 0 and output be posted else error count will increment and no output will be posted (Pipeline stalls continues).The health flag should be "0100".

Permanent Fault Condition: If the transient fault persists for 5 consecutive clock cycles in other words, if error count becomes '5 then we are considering that fault in adder circuitry will not be recovered and declare permanent fault condition. To identity the faulty adder, in the 6th clock cycle the recomputation step will be activated and the inputs to the adders will be $\{(Complement\ of\ OP1) + 1\}$ i.e. First Operand and $\{Complement\ of\ OP2\}$ i.e. Second Operand. The fault handling logic will check the correctness of the output of the recomputation step. The output of recomputation step should be the complement of the output of normal step, therefore the current output of adder1 and the data1 (stored in register1, previous cycle data) will be check for complement relationship similarly the current output of adder2 and the data2 (stored in register2, previous cycle data) will be check for complement relationship. If there any stuck at faults exist in adder circuitry due SEE effects it will be known.

The faulty adder will be permanently isolated and subsequently output will be posted from one

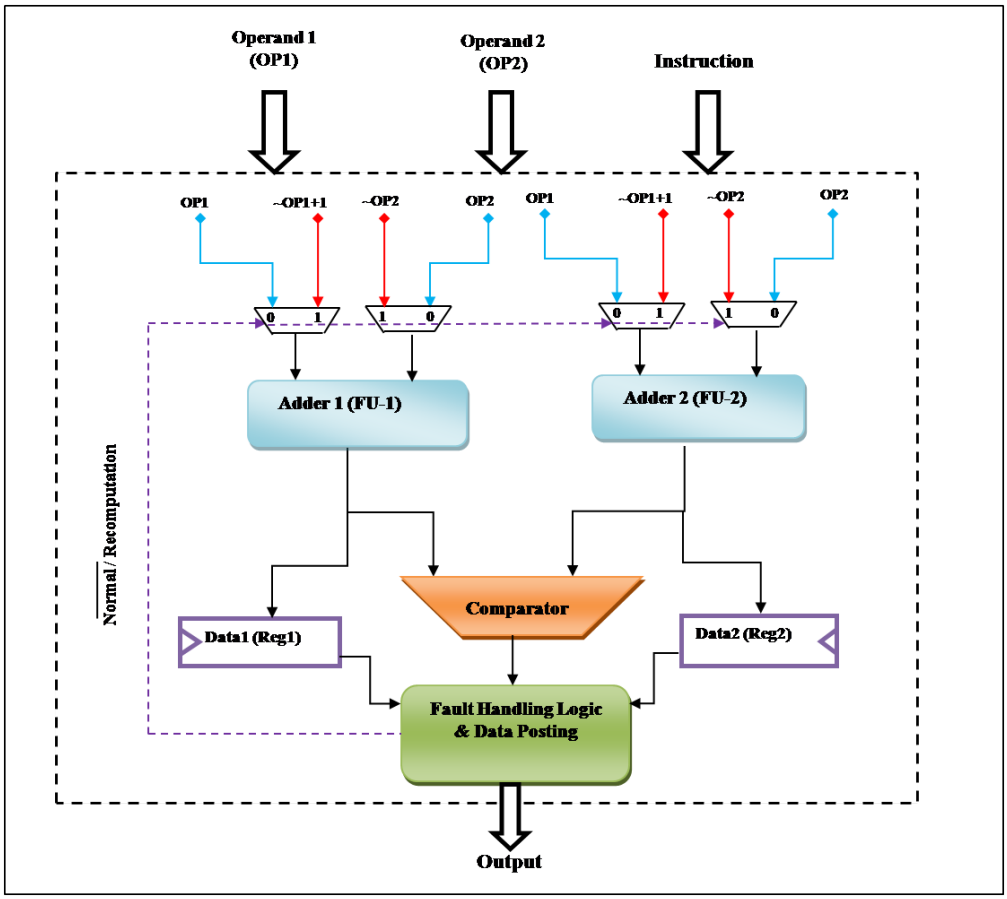


Figure 3.4: Block Diagram of Fault Tolerant Adder

available adder module. The health flag should be any one of them: $\{1100, 1101, 1110, 1111\}$. **Fault Detection Capability**: As F_n and F_r are complementary, single or multiple stuck at faults will be known.

3.4.3.2 Comparator Functional Unit

Our aim here is to make fault tolerant comparator (ft_slt) so that set less than (SLT) related instructions shall be taken care. For comparison, we can do subtraction of 2 operands and check for the sign bit of their result in other words, if the result of $OP1 - OP2$ is positive than $OP2 < OP1$ else $OP1 < OP2$. Subtraction operation shall be performed using fault tolerant adder by using second operand in 2's complement form. Therefore the basic unit of fault tolerant comparator is nothing but fault tolerant adder. Subtraction operation using adder by using second operand in $2's\ complement$ form is applicable only for signed numbers or

unsigned numbers where $OP1 > OP2$. SLT and SLTI instructions deal with signed numbers whereas SLTU and SLTIU instruction deal with unsigned numbers. To resolve this problem, we made 32-unsigned number into signed number by keeping 33^{th} bit as '0'. Therefore the length of comparator becomes 33. The level of fault tolerance is same but we discussed above in case of addition or subtraction.

3.4.3.3 Multiplier Functional Unit

Multiplication related instructions are handled by this FU. In normal computation, $OP1$ and $OP2$ are given to the signed multiplier circuitry which computes output as

$$F_n = OP1 * OP2$$

In recomputation step, first operand will be $2's\ complement\ of\ OP1$ and second operand will remain unchanged as $OP2$. The output of recomputation step will be

$$F_r = (2's\ complement\ of\ OP1) * OP2$$

Therefore, we expect $F_n = -F_r$ (i.e. $(2's\ complement\ of\ F_r = F_n)$, if no error exists in the signed multiplier circuitry.
It should be noted that unsigned multiplication shall be done using signed multiplier circuitry by zero extend the operands. The conceptual block diagram of fault tolerant multiplier is shown in Figure 3.5

No Fault Condition: The inputs to multiplier are OP1 (i.e. First Operand) & OP2 (i.e. Second Operand). As there are no faults in MUL1 and MUL2 circuitry, the output of both multipliers will match and data of MUL1 is posted. The health flag should be "0000".

Transient Fault Condition: If the outputs of MUL1 and MUL2 circuitry are not matches in current cycle then error count becomes 1 and fault handling logic will not post the output. This leads to stall in the pipeline. The outputs will again be matched in next clock cycle if it matches then error count will reset to '0' and output be posted else error count will increment and no output will be posted (Pipeline stalls continues).The health flag should be "0100".

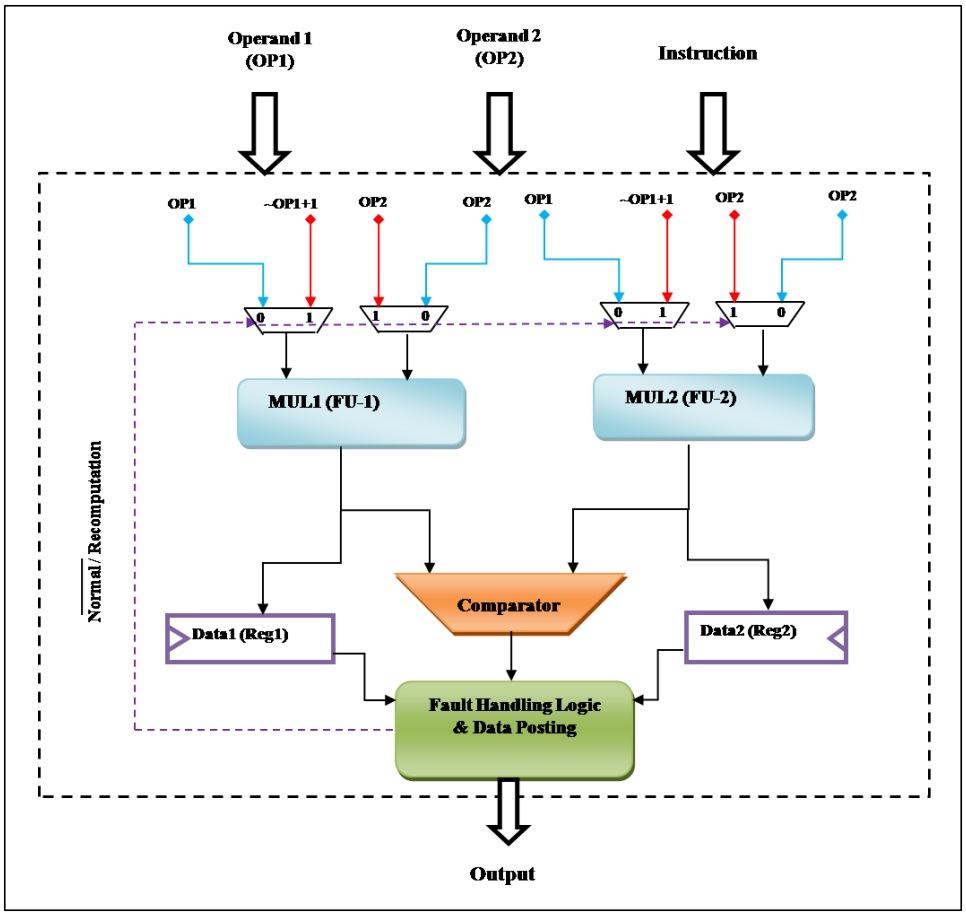


Figure 3.5: Block Diagram of Fault Tolerant Multiplier

Permanent Fault Condition: If the transient fault persists for 5 consecutive clock cycles in other words if error count becomes '5' then we are considering that fault in multiplier circuitry will not be recovered and declare permanent fault condition. To identity the faulty multiplier, in the 6th clock cycle the recomputation step will be activated and the inputs to the multiplier will be $\{2's\ Complement\ of\ OP1\}$ i.e. First Operand and $\{OP2\}$ i.e. Second Operand. The fault handling logic will check the correctness of the output of the recomputation step. The output of recomputation step should be $2's\ complement$ of the output of normal step, therefore the current output of MUL1 and the data1 (stored in register1, previous cycle data) will be check for $2's\ complement$ relationship similarly the current output of MUL2 and the data2 (stored in register2, previous cycle data) will be check for 2s complement relationship. If there any stuck at faults exist in MUL circuitry due SEE effects it will be known.

The faulty multiplier will be permanently isolated and subsequently output will be posted from

one available multiplier module. The health flag should be any one of them: {1100, 1101, 1110, 1111}.

Fault Detection Capability:

1. No fault condition: $F_n = -F_r = 2^n - F_r$
2. Let stuck at failure happens at particular bit position i : Result of normal computation i.e. $R_{normal} = F_n \pm 2^i$ whereas result of recomputation step i.e. $R_{recomp} = F_r$ or $F_r \pm 2^i$. Fault handling logic will do the equality check on R_{normal} and $2's\ complement\ of\ R_{recomp}$. $2's\ complement\ of\ R_{recomp} = 2^n - R_{recomp} = 2^n - (F_r \text{ or } F_r \pm 2^i) = (2^n - F_r)$ or $(2^n - F_r \mp 2^i) = F_n$ or $F_n \mp 2^i$.
   As, $2's\ complement\ of\ R_{recomp} \neq R_{normal}$, error is detected.

3.4.3.4 Shifter Functional Unit

Shift related instructions are handled in this FU. RESO technique has been used here. In normal computation, Operand is given to the shifter which computes output as F_n. In recomputation step, shifted operand (left shifted by 2 bit positions) will be feed to the circuitry.
For n-bit input operand, the length of shifter is $n + 2$ bit. In our case, $n = 32$, therefore, length of shifter is 34 bit. The output of recomputation step is readjusted and compared with the output of normal step which is stored in the register. The mismatch indicates fault in the shifter circuitry.The conceptual block diagram of fault tolerant shifter is shown in Figure 3.6.

No Fault Condition: The inputs to shifter are $OP1$ (i.e. First Operand) & $OP2$ (i.e. Second Operand). As there are no faults in Shifter1 and Shifter2 circuitry, the output of both shifters will match and data of Shifter1 is posted. The health flag should be "0000".

Transient Fault Condition: If the outputs of Shifter1 and Shifter2 circuitry are not matches in current cycle then error count becomes '1' and fault handling logic will not post the output. This leads to stall in the pipeline. The outputs will again be matched in next clock cycle if it matches, then error count will reset to '0' and output be posted else error count will increment and no output will be posted (Pipeline stalls continues).The health flag should be "0100".

Permanent Fault Condition: If the transient fault persists for 5 consecutive clock cycles in other words if error count becomes '5' then we are considering that fault in shifter circuitry will not be recovered and declare permanent fault condition. To identity the faulty shifter, in

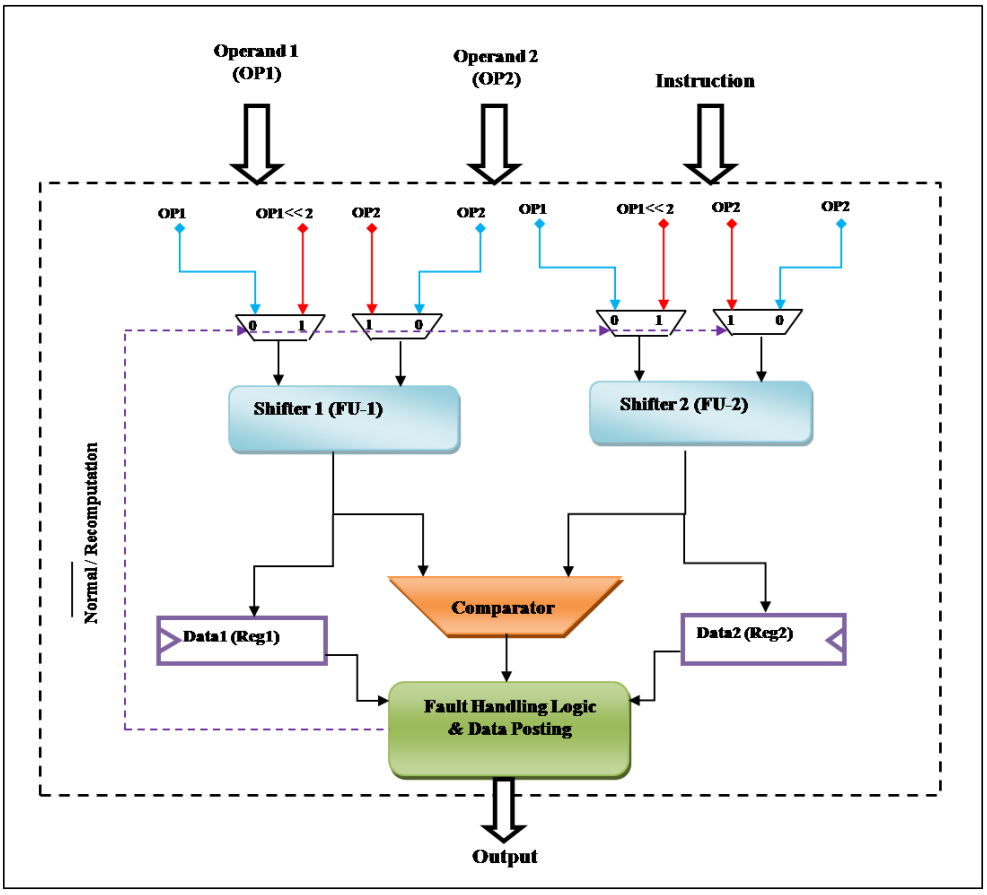


Figure 3.6: Block Diagram of Fault Tolerant Shifter

the 6th clock cycle the recomputation step will be activated and the inputs to the shifter will be $\{OP1\ left\ shifted\ by\ 2\}$ i.e. first Operand and the second operand remains unchanged as $\{OP2\}$ as it is the shift amount. The fault handling logic will check the correctness of the output of the recomputation step. The right shifted output of recomputation step is compared with the output of normal step, therefore the current output of Shifter1 and the data1 (stored in register1, previous cycle data) will be compared similarly the current output of Shifter2 and the data2 (stored in register2, previous cycle data) will be compared. If there any stuck at faults exist in shifter circuitry due SEE effects, it will be known.

The faulty shifter will be permanently isolated and subsequently output will be posted from one available shifter module. The health flag should be any one of them: {1100, 1101, 1110, 1111}.

Fault Detection Capability: Let the output of recomputation step is F_r.

1. No fault condition: $F_n[n-1:0] = F_r[n+1:2]$
2. Let stuck at failure happens at particular bit position i, where $i \in [n-1, 0]$: Result of normal computation i.e. $R_{normal}[n-1:0] = F_n \pm 2^i$, whereas result of recomputation step i.e. $R_{recomp}[n+1:2] = F_n$ or $F_n \pm 2^{i-2}$.
   Fault handling logic will do the equality check on $R_{normal}[n-1:0]$ and $R_{recomp}[n+1:2]$. As, the two results will not match,the error is detected.

3.4.3.5 Logical Functional Unit

logical instructions are handled in this FU. Recomputing with swapped operands technique has been used here. In normal computation, $OP1$ and $OP2$ are given to the logical circuitry which computes output as F_n. In recomputation step, the upper halves and lower halves of operands are swapped. The output of recomputation step is readjusted and compared with the output of normal step which is stored in the register. The mismatch indicates fault in the logical circuitry.The conceptual block diagram of fault tolerant logical unit is shown in Figure 3.7.

No Fault Condition: The inputs to logical circuitry are $OP1$ (i.e. First Operand) & $OP2$ (i.e. Second Operand). As there are no faults in Logical 1 and Logical 2 circuitry, the output of both logical units will match and data of logical unit1 is posted. The health flag should be "0000".

Transient Fault Condition: If the outputs of Logical 1 and Logical 2 circuitry are not matches in current cycle then error count becomes '1' and fault handling logic will not post the output. This leads to stall in the pipeline. The outputs will again be matched in next clock cycle if it matches, then error count will reset to '0' and output be posted else error count will increment and no output will be posted (Pipeline stalls continues).The health flag should be "0100".

Permanent Fault Condition: If the transient fault persists for 5 consecutive clock cycles in other words if error count becomes '5' then we are considering that fault in logical unit circuitry will not be recovered and declare permanent fault condition. To identity the faulty Logical Unit, in the 6th clock cycle the recomputation step will be activated and the inputs to the Logical Unit will be $OP1$ and $OP2$ but with swapped lower halves and upper halves. The

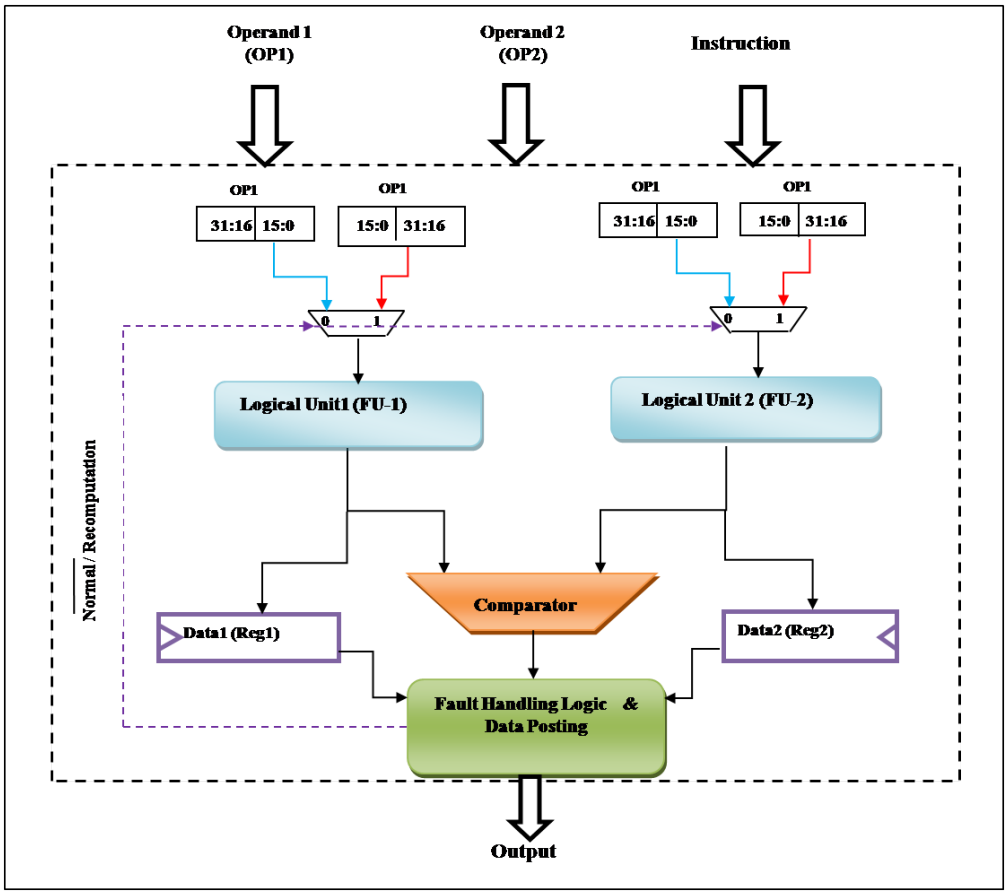


Figure 3.7: Block Diagram of Fault Tolerant Logical Unit

fault handling logic will check the correctness of the output of the recomputation step. The swapped output of recomputation step is compared with the output of normal step, therefore, the current output of Logical Unit1 and the data1 (stored in register1, previous cycle data) will be compared similarly the current output of Logical Unit2 and the data2 (stored in register2, previous cycle data) will be compared. If there any stuck at faults exist in logical circuitry due SEE effects, it will be known.

The faulty logical unit will be permanently isolated and subsequently output will be posted from one available logical module. The health flag should be any one of them: {1100, 1101, 1110, 1111}.

Fault Detection Capability: Let the output of recomputation step is F_r.

1. No fault condition:$F_n[\frac{n}{2}-1:0] = F_r[n-1:\frac{n}{2}]$ and $F_n[n-1:\frac{n}{2}] = F_r[\frac{n}{2}-1:0]$
2. Let stuck at failure happens at particular bit position i : Result of normal computation i.e. $R_{normal} = F_n \pm 2^i$ whereas, result of recomputation step after readjusting i.e.

$$R_{recomp} = \begin{cases} F_n \text{ or } F_n \pm 2^{\left(\frac{n}{2}+i\right)} & 0 \leq i < n/2 \\ F_n \text{ or } F_n \pm 2^{\left(i-\frac{n}{2}\right)} & n/2 \leq i < n \end{cases}$$

   Fault handling logic will do the equality check on R_{normal} and R_{recomp}. As, the two results will not match, the error is detected.

3.5 Fault Tolerant Microarchitecture of 32-bit In-Order RISC-V Processor

3.5.1 Description

We have discussed fault mitigation techniques for memories, registers and combinational logic. Now, we will see where these SEC-DED blocks are placed in the overall architecture and implementation of ALU in dual modular redundant (DMR) configuration. The simplified block diagram of 32-bit fault tolerant processor is shown in Figure 3.8. whereas the detailed 5 stage fault tolerant pipelined processor is shown in Figure 3.9. The instruction memory (IMEM) and data memory (DMEM) are implemented as (32,8) where upper 32 bits are instruction and lower 8 bits are checkbits for corresponding instruction. The functioning of each stage is explained below in brief.

1. **Fetch Stage (IF)**: In this stage, program counter value has been read and passed through SEC-DED module. In this way any error in PC shall be corrected and correct address shall be generated for instruction fetch. The fetched instruction along with checkbits is send to next stage of pipeline. The PC and prediction also moves to next stage.
2. **Decode & Operand Fetch Stage (ID)**: In this stage, the instruction is first passed through SEC-DED module for any correction. The corrected instruction is then decoded for operand fetching and instruction type (i.e. ALU, MEM, Branch, Illegal). The register identity i.e. RS1 and RS2 is passed to operand fetch logic. The value of RS1 and RS2 shall be Architectural Register File (ARF) value or the operand forwarding value from EX or MEM or WB stage. In all cases the value of RS1 and RS2 is in (32,8) format where upper 32 bit is actual data and lower 8 bits are checkbits for that data. This information is then passed to next stage of pipeline through inter stage buffer (ISB).
3. **Execution Stage (EX)**: This is the stage where much of the fault tolerant logic resides. The instruction, operands and program counter are first corrected using SEC-DED module then passed to either ft_alu or ft_memory or ft_branch unit depending upon the instruction type.
   The ft_alu comprises ft_add_sub, ft_multiplier, ft_shifter, ft_logical and ft_comparator units, all are in Dual Modular Redundant (DMR) configuration with fault detection and isolation (FDI) logic. Depending upon the instruction, the operands are appropriately diverted to respective units in the ALU, for e.g. ADD & SUB related instructions will be handled in ft_add_sub unit. The detailed functioning of each unit is already explained. ft_memory and ft_branch uses fault tolerant adder for effective address calculation. The health flag bits are mapped to output pins. If the health flag bits indicate *transient error* condition then the pipeline will be stalled. The operand forwarding path is available from this stage.

4. **Memory Stage (MEM)**: In this stage, data memory (DMEM) is accessed for load and store. In case of load, the data from DMEM is passed through SEC-DED logic for any correction. Appropriate data with checkbits are then passed to next stage for writing to the register file. In case of store, data is written in DMEM at the specified address. Store is not doing anything in writeback stage. The operand forwarding path is available from this stage.

5. **Write Back Stage (WB)**: In this stage, architectural register file (ARF) will be updated. The flush initiated in case of branch misprediction. The operand forwarding path is available from this stage.

3.5.2 Advantages of Proposed Architecture

- Single bit error in the data of IMEM, DMEM, RegisterFile & PC shall be tolerated throughout the pipeline stages.
- The proposed scheme is hybrid scheme where both time redundancy and spatial redundancy are used.
- It can tolerate one FU failure per type in the ALU. In worst case, all together we can tolerate one adder, one multiplier, one shifter, one comparator and one logical unit failure.
- Proposed DMR-ALU with fault isolation saves considerable hardware compare to TMR-ALU and achieve the same level of fault tolerance with time penalty for doing recomputation and fault handling.
- Overall reasonable fault tolerance with acceptable area and timing overhead as compare to base version (no fault tolerance feature).
- Proposed scheme shall be adopted on superscalar class of processors where out of order execution happens at the granularity of FUs in the ALU.

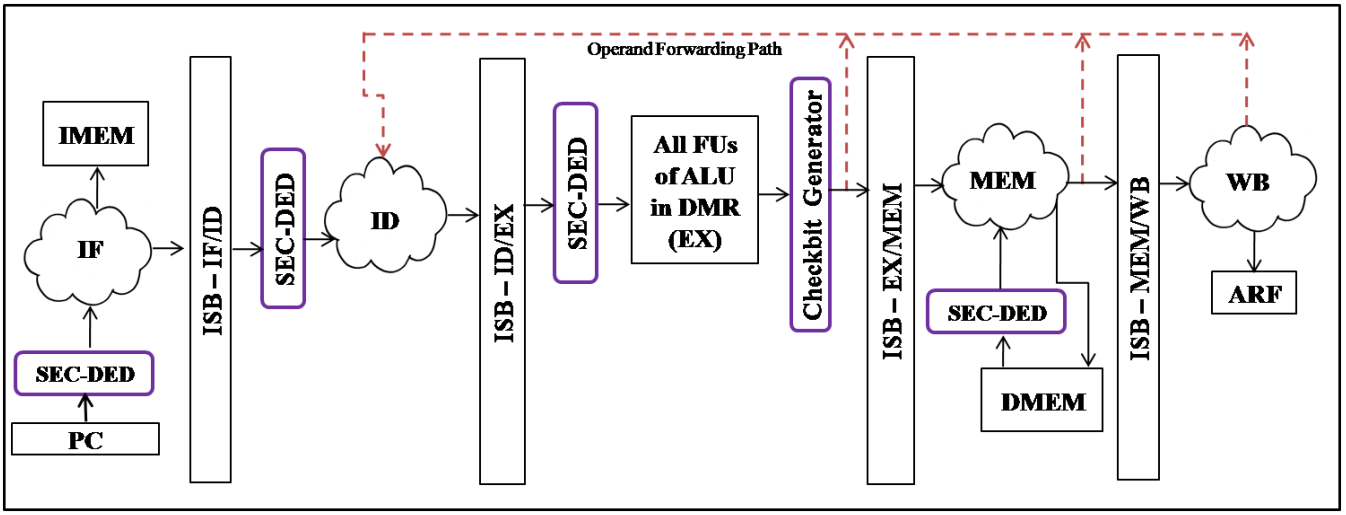


Figure 3.8: Simplified View of Proposed Fault Tolerant Processor Architecture

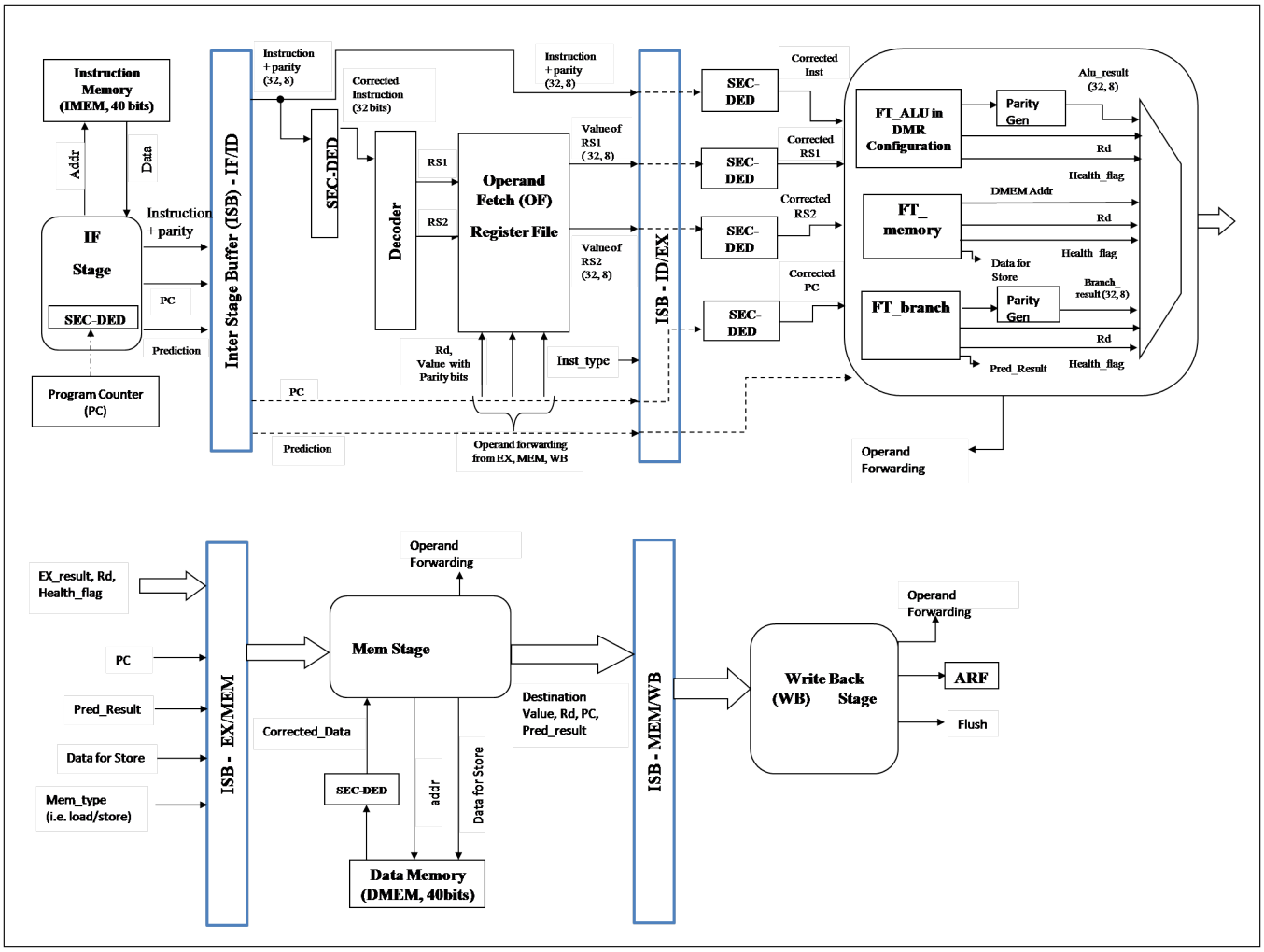


Figure 3.9: Detailed View of Proposed Fault Tolerant Processor Architecture

CHAPTER 4

VERIFICATION PLAN

Verification plays a pivotal role in the system design process. It is intended to check that a product, service, or meets a set of design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of the modelling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications and regulations as time progresses. It is a process that is used to evaluate whether a product, service, or system complies with regulations, specifications or conditions imposed at the start of a development phase. This chapter explains the way,we verified our proposed design.

In our design of fault tolerant version of 32-bit RISC-V Processor, we need intensive and comprehensive verification procedure so that design should able to tolerate faults and no loophole exists .We approached 2 level of verification namely module level verification and system level verification. Following sections we will discuss in details about our verification strategy.

4.1 Module Level Verification

The basic modules involved for regression testing are as follows:

1. Fault tolerant adder
2. Fault tolerant multiplier
3. Fault tolerant logical unit
4. Fault tolerant shifter
5. Fault tolerant comparator
6. Error Correction module (SEC-DED module)

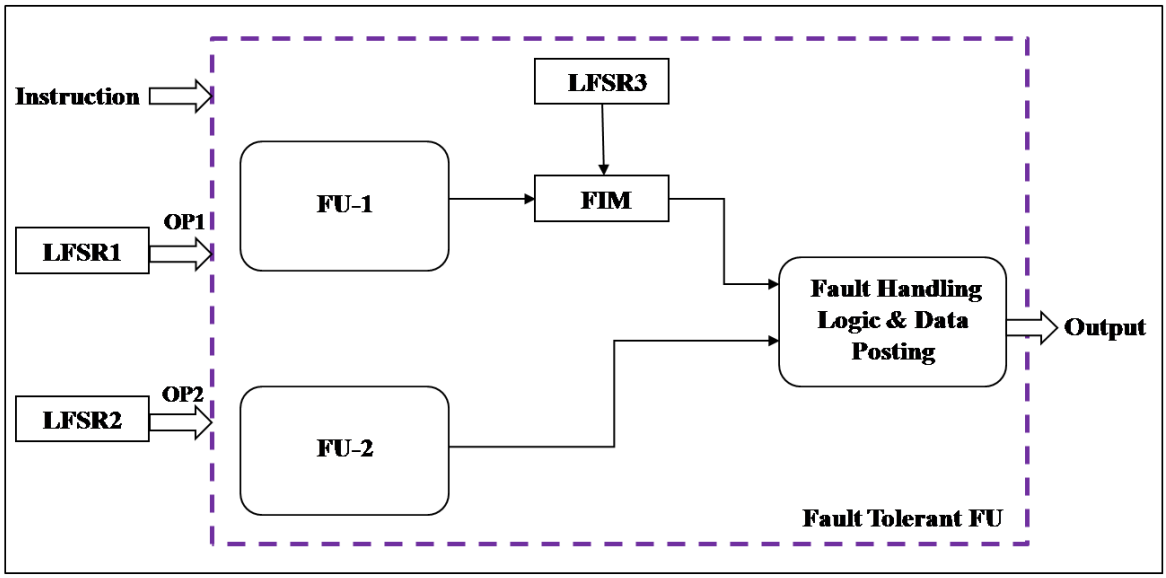


Figure 4.1: Set-up for testing fault tolerant functional unit

4.1.1 Testing of Fault Tolerant FUs

The general setup used for module level testing of above mentioned first 5 modules is shown in Figure 4.1.

Explanation: The FU-1 & FU-2 represents the functional unit in dual redundant configuration. The inputs to the functional unit i.e. operand1 and operand2 are feed through linear feedback shift registers (LFSR). The output of LFSRs are random. Therefore the inputs to functional units are random numbers. The fault injection module (FIM) is placed at the output of FU-1 to simulate runtime errors. FIM is basically doing the ORING of output of FU-1 with the data of LFSR3 for simulating stuck at 1 and for simulating stuck at 0 it is doing ANDING operation. We can either simulate single or multiple bit errors depending upon the value of LFSR3. The simulated error is maintained for 5 consecutive cycles (permanent fault condition) and observes that fault handling logic is able isolate FU-1 and post the data from FU-2. The testing is done in two modes:

1. **Systematic testing**: In this approach, we inject the single bit error systematically one by one from least significant bit to most significant bit. The injected error at particular bit position is maintained for 5 consecutive cycles and observes that fault handling logic is able isolate FU-1 and post the data from FU-2 in all cases. This shows that our recomputing methods are able to detect faulty functional unit. First we simulated stuck at 1 fault at every bit position then we simulated stuck at 0 at every bit position. In all cases

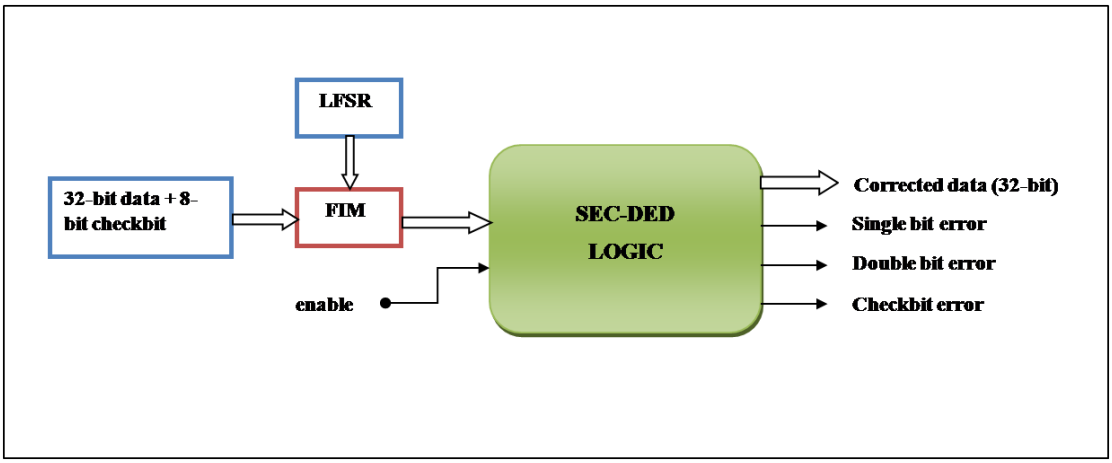


Figure 4.2: Set up for testing SEC-DED Logic

fault handling logic is able isolate FU-1 and post the data from FU-2. The simulation has been carried out for 5000 clock cycles.

2. **Random testing**: In this approach, we inject multiple bit errors randomly at the output of FU-1. The injected errors are maintained for 5 consecutive cycles and observe that fault handling logic is able isolate FU-1 and post the data from FU-2 in all cases. First, we simulated multiple stuck at 1 faults then we simulated multiple stuck at 0 faults. In all cases fault handling logic is able isolate FU-1 and post the data from FU-2. The simulation has been carried out for 5000 clock cycles.

The above testing shows that our recomputing methods are able to detect faulty functional unit.

4.1.2 Testing of Error Correcting Code Logic

The set up of testing error correction module (i.e. SEC-DED module) is shown in Figure 4.2.

Explanation: The inputs to single error correction double error detection (i.e. SEC-DED) block are 32-bit data with corresponding checkbits and enable signal. If $enable\ signal = 1$, then SEC-DED logic is enable otherwise correction logic is disabled and pass the data as such to the output. It is a feature added in case SECDED logic not required.

Fault Injection Module (FIM) is added to simulate runtime errors. FIM is basically doing the XOR of received data bits with the data of LFSR3, therefore inverts the bit and simulate the bit fault. Firstly, we inject the single bit error systematically in 40 bit data (32-bit databits + 8-bit

checkbits). Observe that output of SECDED logic is corrected data with either single bit error asserted or checkbit error asserted depending upon the location of error injection. Secondly, we inject double bit error by setting LFSR and observe that double bit error signal is asserted in all cases.

4.2 System Level Verification

System level verification involves the verification at the processor level. Aim here is to validate the system under different fault rates. Only single bit fault is simulated as in real world situation, SEE effects cause mainly single bit errors.
We have considered the fault injection location as Inter Stage Buffer between decode and execution stage (i.e. ISB-ID_EX) and output of various functional unit like ft_adder, ft_mul etc. used in the design. The ISB has chosen for error injection because everything flows through ISB and injecting fault at ISB-ID_EX may cover the fault of various components of processor like instruction memory, data memory, register file, program counter register etc in the pipeline flow.

4.2.1 Fault Injection at Inter Stage Buffer(ISB)

The fault injection is done on the data of the ID_EX-ISB. This is achieved by placing Fault Injection Module (FIM) just after the ID_EX-ISB. The FIM is basically doing the XOR of the ISB data and the LFSR value. The polynomial used for LFSR is such that it becomes circular shift register with initial feed of 1. In this way we are simulating single bit fault at the ISB data. There is one enable signal input to FIM, if $enable = 1$, then FIM simulates error, otherwise FIM does not simulate error.

4.2.2 Fault Injection at the Fault Tolerant FUs

Both transient and permanent faults are simulated in the ALU. The faults are injected at the output of prime functional unit (i.e. FU-1) and no faults are simulated at the output of redun-

dant functional unit (i.e. FU-2) in dual redundant system. The fault injection module (FIM) is placed at the output of prime functional unit to simulate runtime errors. FIM is basically doing the XOR of output of FU-1 with the data of LFSR, therefore inverts the output bit. The polynomial used for LFSR is such that it becomes circular shift register with initial feed of 1. In this way we are simulating single bit fault though multiple bit errors can also be simulated. There is one enable signal input to FIM, if $enable = 1$ then, FIM simulates error otherwise FIM does not simulate error. The enable bit is taken as one of bit of 8-bit LFSR. If $enable\ bit = 1$, for 5 consecutive cycle then it simulates permanent fault condition otherwise it simulates transient fault condition. In case of permanent fault, the logic will post from redundant functional unit then onwards. This FIM is placed at the output of all prime functional units available in ALU. Figure 4.3 depicts the fault injection at ISB and FUs.

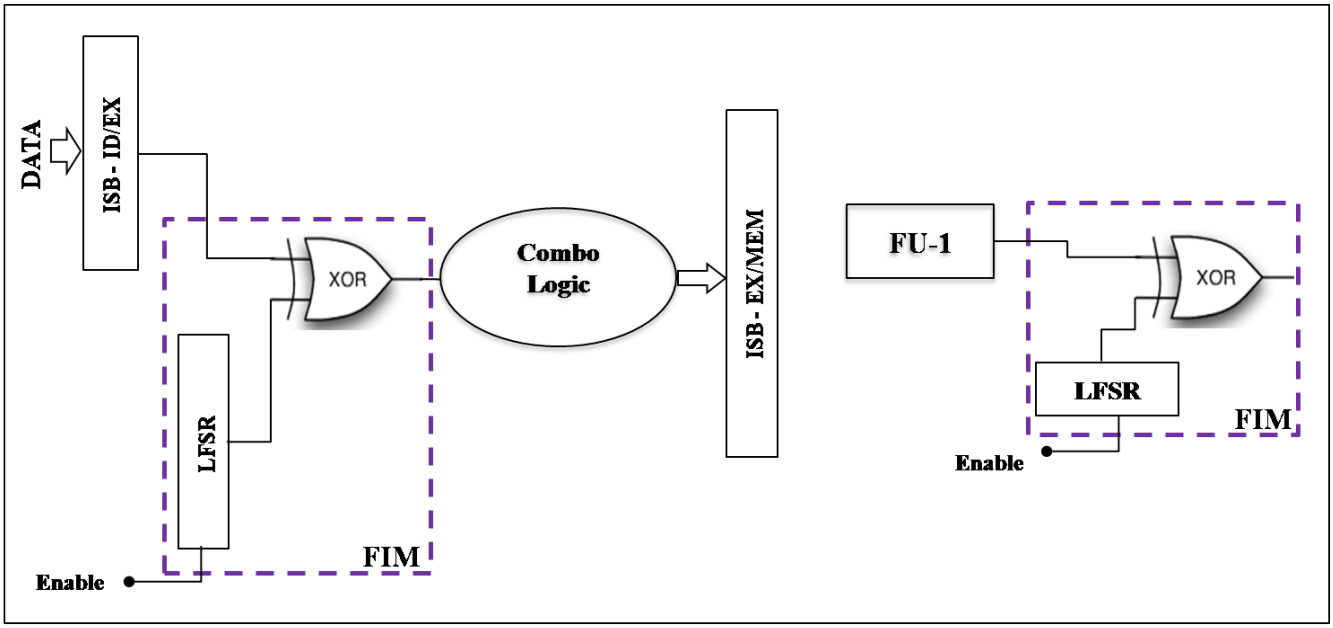


Figure 4.3: Fault Injection at ISB and FU-1

So we have considered 7 fault injection locations in the processor design as mentioned in Table 4.1. The 3 bit LFSR (i.e.lfsr_sel) will randomly determine where to inject errors. The clock to lfsr_sel shall be divided clock (Clk/x), so that a selected location is under fault injection for x clock cycles. One more LFSR (i.e. lfsr_en) is used whose bit will decide whether to inject error or not once a fault injection location is selected for x clock cycles. We can say

that lfsr_sel and lfsr_en will control the enable input of LFSR used for FIM. This scenario can simulate error for $1/2/3/4/5$ consecutive clock cycles. If error persists for 5 consecutive clock cycles, then permanent error flag will be raised and faulty module will be isolated.

S.No	Selected Location	Remarks
1	ISB-ID_EX	Simulating soft/hard error in memories, ARF, PC register
2	ft_adder	Simulating soft/hard error in the adder circuitry used in ALU
3	ft_multiplier	Simulating soft/hard error in the multiplier circuitry
4	ft_shifter	Simulating soft/hard error in the shifter circuitry
5	ft_logical	Simulating soft/hard error in the logical circuitry
6	ft_adder used in EAC for branch related instructions	Simulating soft/hard error in the adder circuitry used in branch module.
7	ft_adder used in EAC for memory related instruction	Simulating soft/hard error in the adder circuitry used in memory module.

Table 4.1: Fault Injection Locations

4.3 Test Set up

The test set up is shown in Figure 4.4. The stimulus is basically the $10,000$ instructions generated using Automatic Test Pattern Generator (ATPG). The selected instruction profile is having Arithmetic instruction- 50%, Multiplication instruction-10%, Branch instruction- 20%,Load and Store instruction- 20%. The golden model (P1) is the reference model with no error injection. The errors are injected for 5000 clock cycles randomly at 7 identified fault injection locations in P2.

The data of 32-internal registers are basically dumped in a file named *register dump file* in the writeback (WB) stage of pipelined processor, similarly the target address and data for data memory (DMEM) corresponds to load and store instructions are also dumped in a file named *data memory dump file*. These dumped files are used for verification. The comparator is basically checking for any mismatch in the register dump files and data memory dump files generated from P1 and P2.

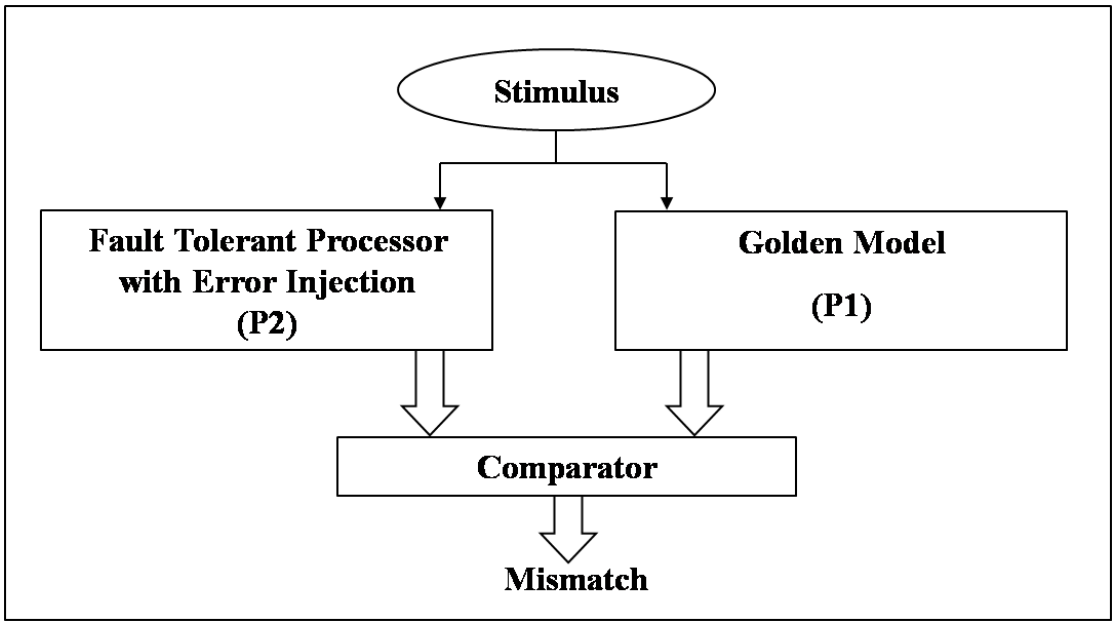


Figure 4.4: Testbench Set Up

CHAPTER 5

IMPLEMENTATION

5.1 Design Flow Overview

The fault tolerant in-order pipelined processor is based on RISC-V ISA. It is completely open-source ISA and suitable for direct native hardware implementation. The entire codes are written in Bluespec System Verilog (BSV) language. The design is verified using bluespec simulator (i.e. bluesim) and verilog simulator (i.e. modelsim). Both ASIC and FPGA implementation have been done for the base processor and the fault tolerant variant, to infer the area, power and timing overhead. The overall flow diagram is shown in Figure 5.1.

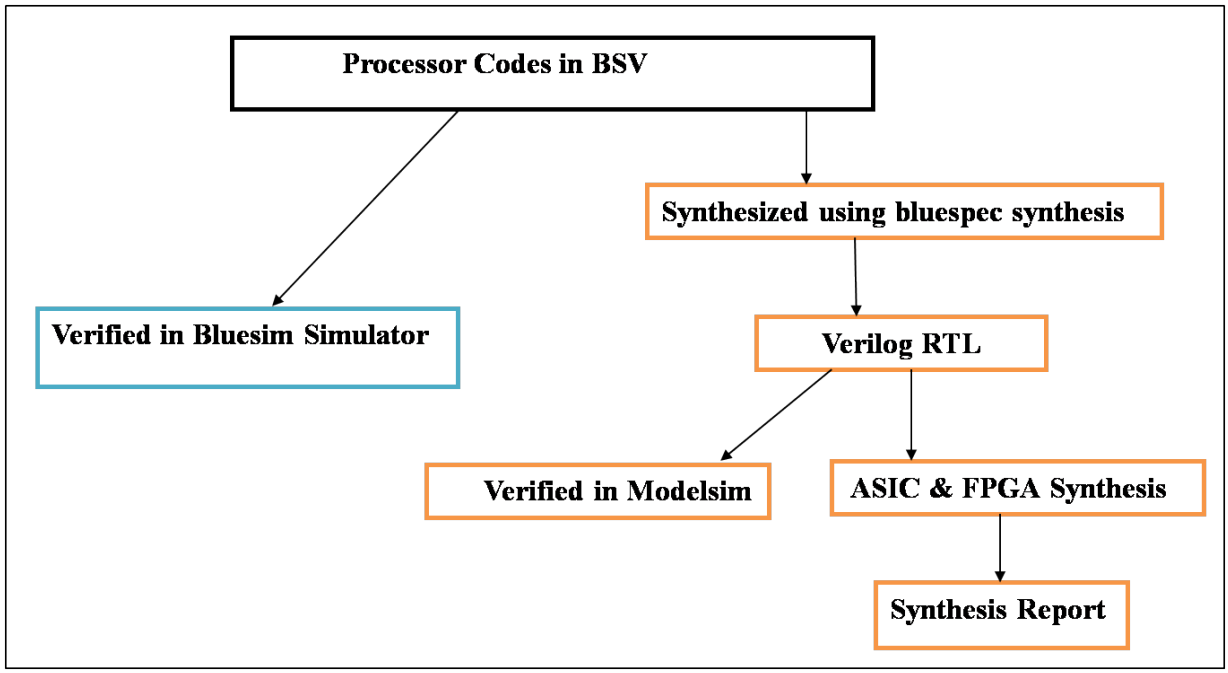


Figure 5.1: Design Flow

5.2 FPGA Synthesis

The design is synthesized to Xilinx Artix family FPGA-XC7A200t-1-FBG676 for area and timing information. The Table 5.1 summarizes the gate utilization and speed of the base processor and fault tolerant one. The reduction in clock frequency in FT processor design is due to the ECC operation on data before feeding to ALU. The combinational delay of SEC-DED logic is 8.74ns.

Processor Version	**Slice Registers**	**Slice LUTs**	**Max Frequency**
Base	2098	4231	55 MHz
Fault tolerant	3047 (+45.2%)	6575 (+55.4%)	36MHz (-34.5%)

Table 5.1: Overhead of the Fault Tolerant Processor (FPGA Synthesis)

5.3 ASIC Implementation

The ASIC implementation of base processor and fault tolerant variant of the processor has been carried out. The designs are targeted for 55 nm technology with 12 track height standard cell and using Low Voltage Threshold (LVT) class of cells. The process,voltage and temperature (PVT) conditions for corner cases are as follows:

Worst Corner Condition (PVT): Slow, $1.08V, 125°$C.

Best Corner Condition (PVT): Fast, $1.32V, -40°$C.

We have used Cadence RTL compiler for synthesis and Cadence Encounter for auto place and route implementation. Table 5.2 summarizes the instance count, power, area, and timing of the base processor and fault tolerant one. The critical path in the base processor is the single cycle multiplier in the EX stage of the pipeline whereas the critical path in the fault tolerant variant is in the EX stage of the pipeline : SEC-DED logic $\rightarrow$ Multiplier$\rightarrow$ Fault handling logic $\rightarrow$ Checkbits Generator. The standard cell area overhead is approximately 69%. The ASIC implementation of base processor and fault tolerant processor is shown in Figure 5.2.

Processor Version	Total Instance Count			Power(mW)			Standard Cell Area (mm^2)	Min. Clock Period (ns)
	Comb.	Seq.	Total	Static	Dynamic	Total		
Base	22542	2634	25176	0.58	20.97	21.55	0.13	2.4
Fault Tolerant	28259	3507	31766	1.16	9.19	10.35	0.2	3.7
Overhead	+25.36%	+33.14%	+26.17%	+100%	-56.17%	-51.97%	+53.8%	+54.17%

Table 5.2: Overhead of the Fault Tolerant Processor (ASIC Implementation)

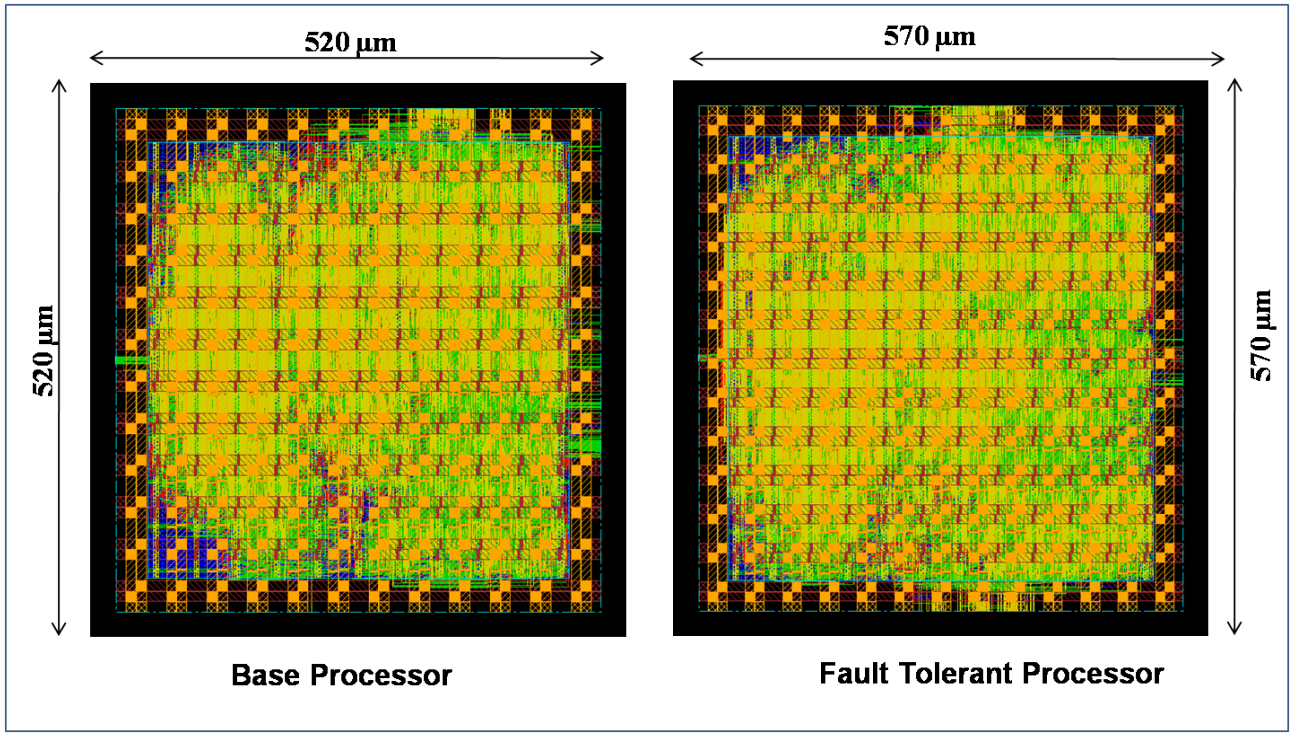


Figure 5.2: ASIC Implementation of Processors

5.4 Performance

The execution time or performance of fault tolerant processor depends upon the error rates, type of error and the location of error. The Table 5.3 provides information about the clock cycle penalty or stall in pipeline due to transient or permanent faults in the system. If the single bit error exists in data of IMEM, DMEM, ARF and PC then it will be corrected online without

any clock cycle penalty and does not pose any stall in the pipeline. Faults in ALU leads to stall in pipeline. If the error in FUs are transient in nature then penalty is depend upon the duration of transient error (TE). If the errors in FUs are permanent in nature then penalty of 5 clock cycles exists. The penalty is high because our assumption for PE condition (i.e.TE persists for 5 consecutive clock cycles). This shall be reduced depending upon the implementation.

Error Type	**Logic Involved for Corrective Action**	**Clock Cycle Penalty**
TE/PE in memories or registers	SEC-DED Logic	0 (No stall)
TE in FUs	Fault handling logic in FUs	$1/2/3/4$
PE in FUs	Fault handling logic in FUs	5

Table 5.3: Clock Cycle Penalty

5.5 Performance Improvement

As, we infer from the Table 5.2 that the standard cell area overhead is 53.8% but the timing penalty is considerable that i.e 54.16%. Our aim to reduce the timing overhead. The following sections discussed the methodologies adopted or to be adopted for performance improvement.

5.5.1 Multi Cycle Execution Stage

The critical path in the fault tolerant processor variant is in the EX stage of the pipeline : SEC-DED logic $\rightarrow$ Multiplier$\rightarrow$ Fault handling logic $\rightarrow$ Checkbits Generator. Therefore, the execution stage is split into 2 stages i.e. EX1 & EX2. In EX1 stage, the SEC-DED has been performed and rest of the thing is carried in EX2 stage. Here the data flowing through the ISB between EX1 & EX2 are vulnerable to soft errors. In other places, the soft error in ISB is tolerated as data flows alongwith checkbits. Therefore, the ISB between EX1 & EX2 should be the hardened against soft errors. The simplified view of improved fault tolerant architecture

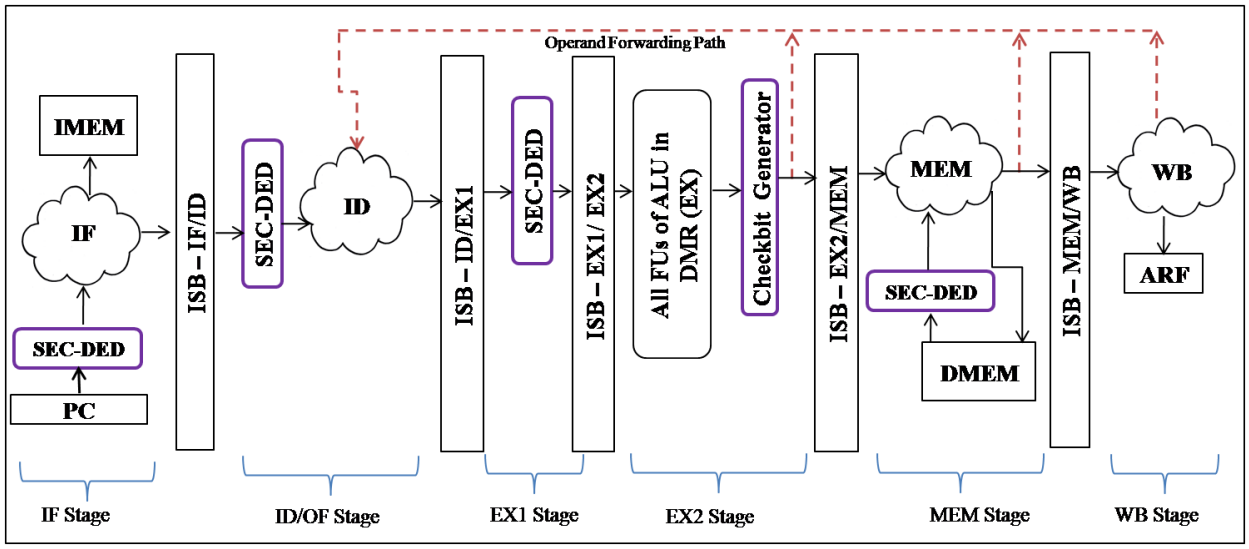


Figure 5.3: Simplified View of Improved Fault Tolerant Architecture

is shown in Figure 5.3. Table 5.4 summarizes the instance count, power, area, and timing of the base processor and improved fault tolerant one.

Processor Version	Total Instance Count			Power(mW)			Standard Cell Area (mm^2)	Min. Clock Period (ns)
	Comb.	Seq.	Total	Static	Dynamic	Total		
Base	22542	2634	25176	0.58	20.97	21.55	0.13	2.4
Improved Fault Tolerant	26458	3638	30096	1.04	10.7	11.74	0.2	3
Overhead	+17.37%	+38.11%	+19.54%	+79.3%	-48.97%	-45.52%	+53.8%	+25%

Table 5.4: Overhead of the Improved Fault Tolerant Processor (ASIC Implementation)

5.5.2 Out of Order Execution

The performance shall be improved by out of order execution where out of order execution happens at the granularity of FUs in the ALU design as shown in Figure 5.4. For e.g. if error exists in multiplier circuitry then non multiplier instruction which are not dependent on previous multiplier instruction shall be executed, meanwhile the corrective action will be taken in the ft_multiplier module. In this way clock cycle penalty shall be reduced. This fault tolerant

ALU shall be easily adopted for superscalar class of processor design (i.e. in-house I-class shakti processor).

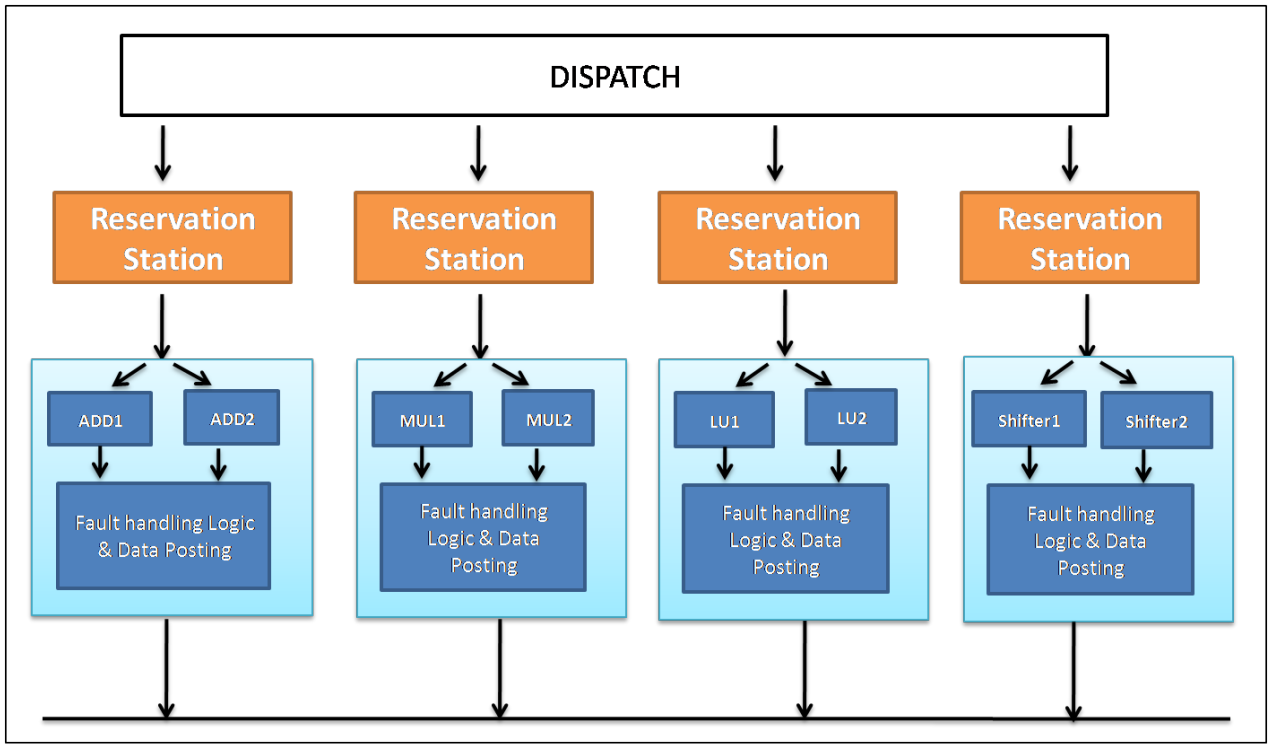


Figure 5.4: Out of Order Execution

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

As CMOS technology moves deep in nanometer range, reliability poses a serious concern in the system design. The probability of soft and hard errors increases, even for ground based system. The processor which needs to work reliably in space or in harsh environment should have techniques to tolerate or mitigate the faults.

This thesis proposed a fault tolerant approach for a 5 stage pipelined processor. The proposed technique was developed with the intention of offering a reasonable fault tolerance with minimal area and power penalty. First we have discussed about the soft and hard errors. The effects of SEUs in memory and combinational logic have been explored. Later the fault avoidance and fault tolerance techniques have been explained. Chapter 3 is the backbone of thesis, where we explained our approach to tolerate soft and hard errors in processor system. Mainly memories,specific registers and ALU have been considered for fault tolerance. The ECC technique used for memories and registers has been explained. The fault tolerant ALU is in DMR configuration where combinational of space and time redundancy techniques have been used to achieve almost same level of fault tolerance as of TMR-ALU therefore reduces hardware and power. Recomputation techniques for FUs have been discussed in detail with their fault detection capability. The verification plan is discussed in chapter 4. Module level and system level verification were discussed. Our results show that we are able to detect and correct single bit errors.

FPGA synthesis and ASIC implementation (55nm technology) of fault tolerant and base version of in-order RISC-V processor have been carried out. Results show that penalty paid in terms of standard cell area overhead is around 60% but on the positive side we achieved a quite reasonable fault tolerance for the system.

6.2 Future Work

Although the results of fault tolerant in-order RISC-V processor presented in this thesis are promising, there are still some directions that can be further improved in future work.

The first direction may be to include DMR-Floating Point Unit in the design and devise a recomputing technique to isolate faulty FPU. Secondly, though the data paths are secured using ECC techniques, control paths are still susceptible to faults. Achieving fault tolerance in control paths shall be looked. Our proposed fault tolerant techniques shall be applied to in-house I-class shakti processor (i.e. superscalar class of processor) where out of order execution happens at the granularity of FUs in the ALU design. This will improve the performance and throughput.

REFERENCES

[1] Janak H.Patel and Leona.Y.Fung, Concurrent Error Detection in ALUs by Recomputing with Shifted Operands, IEEE Transactions on Computers, vol c-31, No 7, July 1982.

[2] Kaijie Wu and Ramesh Karri, Algorithm-Level Recomputing With Shifted Operands A Register Transfer Level Concurrent Error Detection Technique, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol 25, No 3, March 2006.

[3] A.H.Johnston,G.M.Swift and D.C.Shaw, Impact of CMOS Scaling on Single Event Hard Errors in Space Systems,paper published by Jet Propulsion Laboratory,California Institute of Technology.

[4] Chia-Hsiang Chen, David Blaauw,Dennis Sylvester and Zhengya Zhang, Design and Evaluation of Confidence-Driven Error-Resilient Systems,IEEE Transactions on Very Large Scale Integration (VLSI) Systems,vol 22,No 8,August 2014.

[5] Nguyen Minh Huu,Bruno Ribisson,Michel Agoyan and Nathalie Drach, Low-cost fault tolerance on the ALU in simple pipelined processors,IEEE 2010.

[6] I.Wali,A.Virazel,A.Bosoi,L.Dilillo,P.Girard and A.Todri, Protecting Combinational Logic in Pipelined Microprocessor Cores Against Transient and Permanent Faults,IEEE 2014.

[7] Fan Wang, Vishwani D. Agrawal, "Single Event Upset: An Embedded Tutorial", VLSID, 2008, , 2008, pp. 429-434, doi:10.1109/VLSI.2008.28

[8] Shekher Borkar, Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation, IEEE Computer Society, 2005.

[9] Shekher Borkar, "Design challenges of technology scaling," Micro, IEEE , vol.19, no.4, pp.23,29, Jul-Aug 1999.

[10] Simon Tam, Xilinx application note on single error correction and double error detection, XAPP645 (v2.2), August 9, 2006.

[11] Hai Yu, Thesis on Low-Cost Highly-Efficient Fault Tolerant Processor Design for Mitigating the Reliability Issues in Nanometric Technologies,2nd December,2011.

[12] Chris Weaver,Todd Austin, " A fault tolerant approach to microprocessor design", Appears in Dependable Systems and Networks(DSN),July 2001.

[13] Bluespec Inc, Bluespec SystemVerilog Reference Guide Revision: 30 January 2012.

[14] University of California, The RISC V Instruction Set Manual, Ver2.0, May 6, 2014.