UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Cost-Effective Radiation Hardened Techniques for Microprocessor Pipelines

by

Yang Lin

Thesis for the degree of Doctor of Philosophy

April 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Doctor of Philosophy

COST-EFFECTIVE RADIATION HARDENED TECHNIQUES FOR MICROPROCESSOR PIPELINES

by Yang Lin

The aggressive scaling of semiconductor devices has caused a significant increase in the soft error rate induced by radiation particle strikes. This has led to an increasing need for soft error-tolerance techniques to maintain system reliability, even for sea-level commodity computer products. Conventional radiation-hardening techniques, typically used in safety-critical applications, are prohibitively expensive for non-safety-critical microprocessors in terrestrial environments. Providing effective hardening solutions for general logic in microprocessor pipelines, in particular, is a major challenge and still remains open. This thesis studies the soft error effects on modern microprocessors, with the aim to develop cost-effective soft error mitigation techniques for general logic, and provide a comprehensive soft error treatment for commercial microprocessor pipelines.

This thesis presents three major contributions. The first contribution proposes two novel radiation-hardening flip-flop architectures, named SETTOFF. A reliability evaluation model, which can statistically analyse the reliability of different circuit architectures, is also developed. The evaluation results from 65nm and 120nm technologies show that SETTOFF can provide better error-tolerance capabilities than most previous techniques. Compared to a TMR-latch, SETTOFF can reduce area, power, and delay overheads by over 50%, 86%, and 78%, respectively. The second contribution proposes a self-checking technique based on the SETTOFF architectures. The self-checking technique overcomes the common limitation of most previous techniques by realising a self-checking capability, which allows SETTOFF to mitigate both the errors occurring in the original circuitry, and the errors occurring in the redundancies added for error-tolerance. Evaluation results demonstrated that the self-checking architecture can provide much higher Multiple-Bit-Upsets-tolerant capabilities with significantly less power and delay penalties, compared to the traditional ECC technique, for protecting the register file. The third contribution proposes a novel pipeline protection mechanism, which is achieved by incorporating the SETTOFF-based self-checking cells into the microprocessor pipeline. An architectural replay recovery scheme is developed to recover the relevant errors detected by the self-checking SETTOFF architecture. The evaluation results show that the proposed mechanism can effectively mitigate both SEUs and SETs occurring in different parts of the pipeline. It overcomes the drawback of most previous pipeline protection techniques and achieves a complete and cost-effective pipeline protection.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my brilliant supervisor, Professor Mark Zwolinski, for his support and invaluable guidance through my PhD. His vision and wisdom in approaching a problem is the single greatest thing I have learnt from this project. I will try to keep following it and benefit from it for the future of my life.

I would like to take this opportunity to thank Dr. Basel Halak, for his excellent technical advices and insightful discussions on this research. His reviews of this work have always been invaluable, and I am grateful to be able to work with him. Also, I would like to thank Dr. Koushik Maharatna, for his revisions of my work and his valuable advices for my first year and second year reports.

I would also like to offer my thank to the School of Electronics and Computer Science (ECS), for supporting my work by means of studentship and state of the art research facilities. This PhD project would not have been possible without this support. My thank also goes to all my colleagues who have helped me in many ways during my PhD. The joyful and active research atmosphere that is created by them will always be in my memory.

Finally, my parents and my girlfriend have always been my strongest supporters. I am greatly thankful for their love and faith on me, and I would like to dedicate this work to them.

Chapter 1

Introduction

Since the invention of the CMOS in 1970, the technology has been progressing rapidly. The demands for the performance, functionality and power-efficiency have been increased considerably for semiconductor devices. Microprocessors, in particular, are the main driver of this trend. In 1965, Gordon Moore of Intel Corporation proposed Moore's Law [1], which predicted that the number of transistors integrated in a single chip will double every 18-24 months. Over the past few decades, the drastic scaling of the feature size has lead to an exponential growth in the number of transistors, which has significantly improved the functionality of the chips. However, along with technology advance, each succeeding technology generations have introduced new design challenges, due to the limited budgets for chip area, energy dissipation, and the increasing requirement for performance, functionality, and reliability. Innovating solutions need to be carried out for satisfying various design constraints simultaneously during the design process. The complexity of finding a balance between these constraints to meet all the specifications has been increased substantially.

The energy consumption of a chip, for instance, has recently become one of the highest priorities among various design constraints due to the increased leakage power dissipation and the rapid extension of the mobile device market. The reliability issues caused by the hard errors induced by permanent faults, and the soft errors induced by transient faults have also become a significant challenge. Hard errors can typically be addressed during the fabrication process, but addressing soft errors requires appropriate detection and correction mechanisms. Therefore, the reliability issues caused by soft errors have become the major cause of concern.

This thesis describes the effects of radiation-induced soft errors on modern integrated circuits, and proposes new cost-effective techniques to mitigate soft errors and improve system reliability. The proposed techniques have been implemented in modern technology nodes and validated on a RISC microprocessor.

This chapter briefly introduces the preliminary background about the radiation-induced soft errors, and outlines the work described in the subsequent chapters of the thesis. Section 1.1 and Section 1.2 introduce the basic concept of soft errors and their impact on digital circuits, respectively. Section 1.3 describes the types of soft errors, and the metrics to calculate soft error rates. The technology trends on soft error effects are described in Section 1.4. Section 1.5 provides an overview of the existing soft error mitigation techniques. The motivations for the research are introduced in Section 1.6, and the thesis organisation is described in Section 1.7.

1.1 Radiation-Induced Soft Errors

The first report of problems caused by soft errors at sea-level traces back to 1978. Soft errors were found in the DRAMs of Intel computer chips. Intel Corporation declared their problem to be soft errors induced by alpha particle contaminations [2]. Subsequently, IBM Corporation encountered soft errors affecting their chips due to cosmic radiation in 1984 [3]. In 2000, Sun Microsystems observed soft error problems in their commercial products. The SRAM chips of their UltraSPARC-II-based server were corrupted, due to the insufficient soft error-mitigation features incorporated in the chips [2].

Soft errors are a manifestation of underlying transient faults. There are variety of sources that can cause transient faults, such as transistor variabilities, thermal cycling, and radiation particle strikes, etc [4]. This thesis focuses on the radiation-induced transient faults caused by two sources – alpha particles from the packaging and bonding materials, and the neutrons from the atmosphere and beyond.

Alpha particles are produced from the impurities existing in the packaging materials, and are very difficult to completely eliminate. Typically, alpha particles only possess the kinetic energies of a few Mev, but they can affect semiconductor devices by depositing a dense track of charge and create electron-hole pairs as they pass through the bulk or substrate of a transistor [4]. A sufficient number of electron-hole pairs can result in transient current pulses, which induce transient faults. Neutrons are one of the subatomic particles that construct atoms. Neutrons are induced by cosmic rays interacting with the earth's atmosphere [5]. Unlike alpha particles which interact directly with electrons, neutrons create transient faults in an indirect manner: they interact with silicon, oxygen, or other atoms to cause the emission of secondary particles such as protons, alpha particles, neutrons, etc. These secondary particles can force ionization tracks to create a sufficient number of electron-hole pairs, which induce transient faults in semiconductor devices [4].

Certain shielding techniques, such as using small amounts of epoxy or nonradioactive lead, can provide a shield against alpha particles and therefore reduce the circuit interaction with such radiations. However, alpha particles are difficult to completely eliminate. A small number of alpha particles can still seriously affect chip operations [4]. Neutrons, on the other hand, cannot be shielded unless by using thick concrete, which is not practical. Consequently, detection and correction techniques are essential in order to tackle the faults induced by these particles.

1.2 Impact of Radiation Particle Strikes: Transient Faults

In order for particle strikes to cause malfunctions in the circuit, the charge produced by the particles has to exceed a certain threshold. The occurrence of the radiation-induced circuit malfunction depends on two factors, namely, the critical charge of the circuit, and the charge collection efficiency. The critical charge of the circuit, represented by Q_{crit}, is defined as the minimum charge required to cause circuit malfunctions. Q_{crit} can typically be derived through circuit model simulations, by injecting different currents into the circuit until it fails. The charge collection coefficient, denoted by Q_{coll}, is the ratio of the charge collected by the circuit and the charge generated by the particles. If the collected charge crosses Q_{crit} when a circuit node is struck by radiation particles, different transient faults will arise according to the types of the node being attacked. If the node belongs to a storage cell, such a RAM, a latch, or a flip-flop, the state stored in the cell may be flipped, resulting in a Single-Event-Upset (SEU). If the node belongs to a combinational cell, it may produce a transient voltage pulse named a Single-Event-Transient (SET).

SEUs are typical transient faults manifested at the output of storage cells, which may incur soft errors directly. SETs, however, might not affect the system as they may be masked during the propagation through the circuit. Only the SETs that are captured by the forward storage cells can cause an error. There are three commonly observed masking mechanisms: logic masking, electrical masking, and latch-window masking [6] [7] [8]:

- Logic masking: an SET pulse can be logically masked when it appears at the portion of the circuit that does not affect the circuit output. In a two-input AND-gate, for instance, if the first input is zero, the second input can be regarded as 'don't care'. This is because no matter whether the second input is zero or one, the output of the AND-gate is zero. An SET occurring at the second input of the AND-gate will be logically masked if the first input is one.
- Electrical masking: the electrical masking can happen when an SET pulse attenuates during the propagation before it arrives the forward storage cells. The storage cells, in this case, will not capture the SET.

- Latch-window masking: Latch-window masking occurs when the SET pulse does not arrive at the forward storage cells (latches or flip-flops) during their sampling windows. The sampling window of a flip-flop, for example, is the clock transition. The SETs that do not arrive at its input during the clock transitions will be masked.

1.3 Types of Soft Errors and Metrics

The radiation-induced SETs and SEUs arising from circuit nodes can result in visible soft errors if they are unmasked during propagations. Soft errors can be defined at different levels, depending on how the transient faults affect the outcome of circuit operations and how the system is being used. Architecturally, soft errors are typically categorised into two types: Silent Data Corruption (SDC) and Detected Unrecoverable Error (DUE) [4]. SDCs and DUEs are the soft errors defined at the highest scope which corrupt the user interface of a system. As the name suggested, SDCs are the data corruptions that affect the user without any warnings or error flags. In contract, a DUE is an error detected by the system, but recovery is not possible. DUEs can crash the system but can avoid data corruption. SDCs are therefore normally perceived as significantly more harmful than DUEs.

The soft error rate is often expressed by two metrics, Mean Time To Failure (MTTF) and Failure In Time (FIT) [4]. MTTF of a electronic component expresses the mean time elapsed between the component encountering two errors. FIT is defined as an error in a billion (10^9) hours. Equation 1.1 explains the relationship between the MTTF and FIT.

$$\text{MTTF in years} = \frac{10^9}{\text{FIT rate} \times 24 \text{ hours} \times 365 \text{ days}} \tag{1.1}$$

The MTTF or FIT of the components constituting a system can be used to derive the MTTF or FIT of a whole system. The SDC and DUE rates of a system can be separately expressed. For instance, a system can have MTTF SDC or FIT SDC, it could also have MTTF DUE or FIT DUE.

1.4 Soft Error Effect on Technology Scaling Trends

The reliability problems caused by soft errors used to be a cause of concern merely in space applications. This is because space applications are safety-critical, and they routinely encounter heavy radiation hits generated by primary and secondary cosmic rays in space [4]. However, drastic technology scaling, increasing system complexity,

supply voltage reduction, and increasing operation speed have dramatically increased the soft error susceptibilities of modern ICs. The soft error effect has become a major cause of concern even for electronics in terrestrial environments. This section describes the prediction of future scaling trends based on current research and knowledge.

The Soft Error Rate (SER) can be predicted by the model proposed by Hazucha and Svensson [9], shown as Equation 1.2 :

$$\text{Circuit SER} = \text{Constant} \times \text{Flux} \times \text{Area} \times e^{-\frac{Q_{crit}}{Q_{coll}}} \tag{1.2}$$

Constant is a parameter that depends on the technology process and the circuit design style. The Flux is the flux of alpha particles or neutrons at a particular location. Area represents the area of the circuit that is sensitive to radiation hits. This model can illustrate the general SER trend over process technology. Borkar, et al. [10] showed that each time the technology progress to a new generation, the transistor size and the supply voltage is reduced by 30%. The shrinking feature size makes the area occupied by an equivalent circuit (such as a flip-flop) go down. This leads to less area to be exposed to radiation particles, which will decrease the SER. However, according to the equation, charge = capacitance × voltage^2, Q_{crit} will be noticeably decreased across process generations. This is because the supply voltage is reduced, and the capacitance is decreased due to the shrinking transistors. The charge that is required for the particles to incur transient faults, i.e. the Q_{crit}, will therefore be reduced. The decreasing Q_{crit} will increase the vulnerability of the circuit and thus increase the SER over generations.

For latches and flip-flops, the two effects seem to have cancelled each other out in present progress technologies. Therefore, the SER for a single cell stays roughly constant over the past few generations [4]. The SERs for the combinatorial logic gates are typically lower than that for latches because many of the transient faults are masked by the masking mechanisms discussed in Section 1.2. However, the SERs for combinational gates show a different scaling trend, which has increased across process generations. The reason for this can be explained by the changing of the masking mechanisms, as follows:

The logic masking effect will remain constant across technology generations. Electrical masking, nevertheless, will decrease with technology scaling. This is because the delay of the gates will keep reducing, which will allow more transient pulses to propagate without attenuations. In addition, today's microprocessors apply a high degree of pipelining with increasing operating frequencies. This means that the clock period is decreasing, and more flip-flops or latches are inserted between pipeline stages. Both of the factors will significantly decease the probability of the transient pulses being masked by the latching-window. Shivakumar et al. [8] predicted that the SER caused by the SET pulses in the combinatorial gates will increase exponentially, and will become close to or even exceed that is caused by the latches in 65nm technology or below. Recently, Mahatme, et

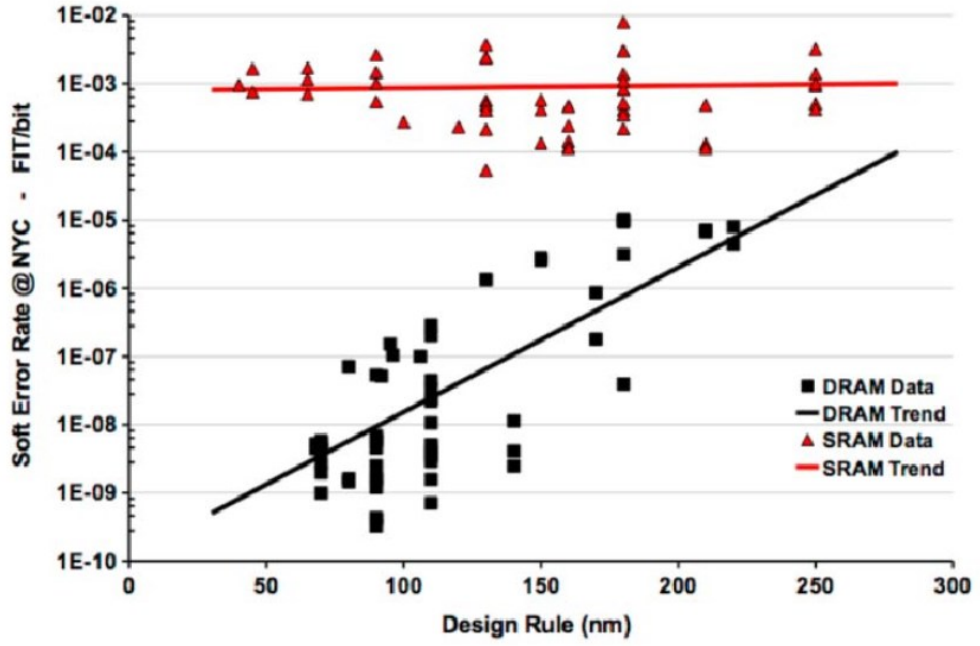

Figure 1.1: The SRAM and DRAM soft error rate trends per bit vs. technology process [12].

al. have identified and reported the operating frequencies at which the SER from the combinational circuits exceed that from the flip-flops in 28nm CMOS technology [11].

For the SRAMs which are typically constructed by six transistors making up two cross-coupled inverters, the Q_{crit} is sufficiently low compared to general logic [1]. As a result, the shrinking area dominates the impact on SER. This is commonly referred to as the saturation effect. Nevertheless, the saturation effect does not indicate that the SER of an individual SRAM cell will decrease with technology scaling. Figure 1.1 shows the SER scaling trend for SRAM per bit, which is obtained in the high energy neutron flux at New York City [12]. The SER of a single bit SRAM remains roughly constant, with a slight decrease from 250nm down to 50nm technology. This is because in the 'saturation' region, the Q_{crit} is low enough to cause the circuit to be sensitive to any particle strikes. The effect of the shrinking area, is therefore mostly balanced out. Figure 1.2 shows the SER trend of the SRAM at system level. Since the number of SRAMs bits packed on a chip has kept increasing across generations, the system-level SER shows an increasing trend.

In contrast, as shown in Figure 1.1 and Figure 1.2, the SER of the DRAMs manifests an optimistic trend. This is because techniques such as adding more stacks, using thinner dielectrics, and using bigger capacitors have been found which can decrease the collected charge without affecting the Q_{crit}. The single bit SER for DRAMs has

[1] According to Mukherjee, [4], the Q_{crit} of SRAMs is usually 5-10 times lower than that of latches.

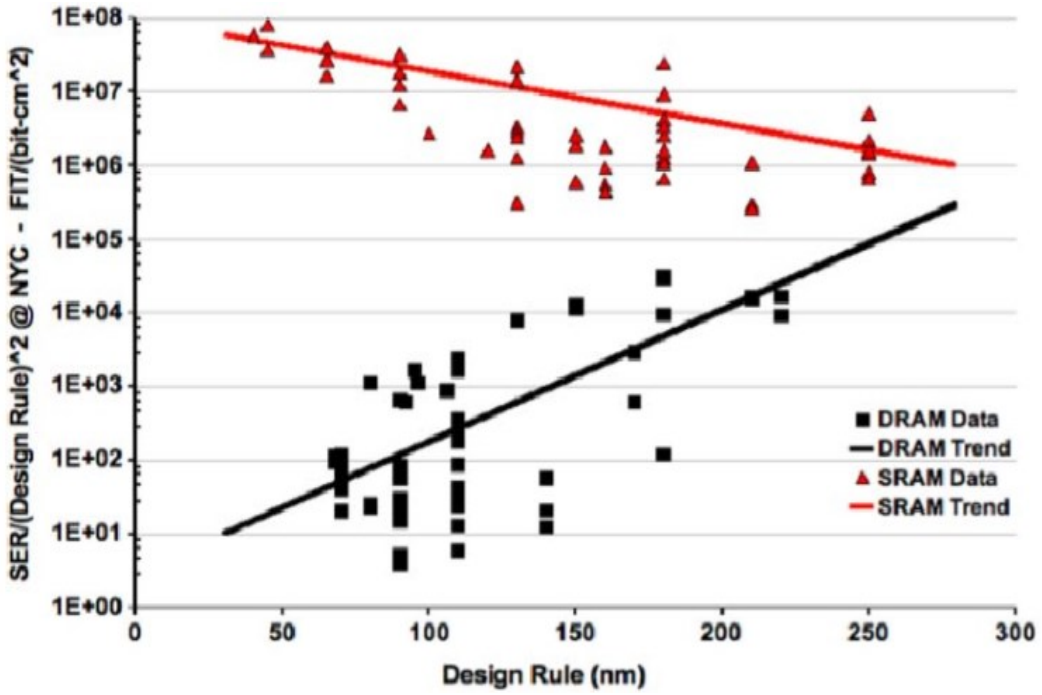

Figure 1.2: The SRAM and DRAM soft error rate trends per chip vs. technology process [12].

decreased exponentially over the past few generations. The system-level SER for DRAMs has also been decreasing. However, the overall SER of DRAMs at the system level still has a big impact on the reliability due to the large number of DRAMs produced.

On the other hand, radiation particles can also cause multiple transient faults in memories and logic. This is normally referred to as Multiple-Bit-Upsets (MBUs). There are two types of MBUs: Spatial MBUs which are caused by the charge sharing between adjacent circuit nodes induced by a single particle strike [13] [14], and temporal MBUs, which are the MBUs arisen from multiple particles simultaneously striking multiple nodes. As chips are becoming more compact, the reliability problems caused by MBUs are getting increasingly serious, especially in RAMs [15] [16] [17] [18] [8]. Raine, et al. showed that the probability of the MBU occurrence in 65nm technology is fairly small, but when it goes down to 45nm and 32nm technologies, the MBU probability in SRAMs increases considerably, and even exceeds the SBU probability [19]. In addition, it has also been suggested that supply voltage scaling also triggers noticeable increases in MBU rates in modern CMOS technology [20] [21].

Overall, the SER of a system can be approximately predicted as the function of the SER caused by the unprotected logic and RAMs. As the system complexity is increasing and the total number of transistors is doubling every generation, the number of unprotected cells will also double, unless more aggressive error-tolerance features are incorporated into the system. Since the SERs of the individual cells generally remain constant, we

can expect that the overall SER of a system will roughly double in every new generation [4]. Another observation is that, although the SRAMs and DRAMs used to be the major contributors of the SERs in a system, the contribution of the SERs in general logic is catching up. This is because in modern microprocessors, the amount of the chip area occupied by the general logic are increasing with chip complexity, thus even a small SER in a logic gate could cause big impacts to the overall system SER. SRAMs and DRAMs can typically be protected by using Error Correction Codes (ECCs) effectively. Effective protection techniques for general logic are still open. General logic is much harder to protect by simple coding techniques, therefore hardware redundancies or radiation hardened cells are normally required. An overview of the current existing soft error mitigation techniques will be carried out in the following section.

1.5 Soft Error Mitigation Techniques and Their Limitations

Radiation-induced soft errors have long been managed by various techniques at different abstraction levels in electronics. At device level, techniques are typically used to increase the critical charge of the circuit by using bigger capacitance, or to minimise the amount of collected charge on a device node under radiation hits. At circuit level, radiation hardened cells are designed, for instance, by duplicating the state-holding elements in a latch [22]. At gate-level or architectural level, techniques such as Triple-Modular-Redundancy (TMR) or Dual-Modular-Redundancy (DMR) can be applied to combat soft errors using redundant components. There are also techniques at software level which reply on duplicating the execution of programs to provide error resilience [23].

Coding techniques such as ECC and parity coding are commonly effective solutions for the soft errors occurring in SRAMs and DRAMs. ECC adds information redundancies (ECC bits) to identify and correct bit-flip errors through particular encoding and decoding algorithms. The ECC bits need to be calculated or decoded during each memory access operation, which can induce certain performance and power consumption overheads. In comparison, parity coding is a cheaper solution which can only detect errors, but cannot correct. In most circumstances, vendors are able to adapt these coding techniques to the computer memories to achieve various reliability requirements with acceptable overheads.

The ECC or parity coding techniques, however, are very hard to use for protecting general logic against particle strikes. This is because the sequential elements, such as flip-flops and latches, are distributed across the entire chip. Hardware redundancy, such as TMR, is a conventional protection scheme for logic in high-end or safety-critical electronic applications such as aircraft. Applying TMR is expensive, as it requires over 200% extra area and power. The reliability requirements in these applications are

sufficiently high such that the large overheads incurred by the error-mitigation features can be tolerated. Nevertheless, most of these conventional solutions are far too expensive for protecting products in mainstream commodity computer markets.

Another typical solution for soft errors in general logic is using radiation hardened sequential cells. Among general logic gates, sequential elements, such as flip-flops and latches, have the highest priority for protection against particle strikes. Evaluation results presented by Ebrahimi have shown that apart from the SER produced by the memories, the flip-flops and latches contribute most to the system SER in a microprocessor [24]. This is because: 1. the flip-flops and latches can produce SEUs, which have a higher probability to corrupt the system compared to the SETs from combinatorial gates. 2. Only the SETs that are captured by the sequential elements are dangerous to the system. Therefore, hardening the sequential cells can potentially protect the system against both SETs and SEUs. One the other hand, the number of sequential elements in a system is much smaller than that of combinatorial gates. Protecting the sequential gates can be easier and more efficient. However, most of the previous radiation-hardened sequential cells are either too expensive, or cannot provide decent tolerant-capabilities for both SETs and SEUs. Few of them can effectively achieve a complete protection for microprocessor pipelines.

Another limitation of most previous techniques is that they are not self-checking. The soft error-mitigation features in previous techniques are normally achieved by adding redundancies. However, most previous techniques can only mitigate the errors occurring in the original circuitry using redundancies, but cannot protect the redundancies themselves. The added redundancies increase the area and thus increase the soft error vulnerability of the whole circuitry. The errors occurring in the redundancies may still corrupt the cell.

A detailed overview of previous soft error-mitigation techniques is in Chapter 2.

1.6 Research Motivations and Objectives

The design of modern commercial microprocessors needs to meet the required SER budget while still being competitive in the market. As the soft error issue keeps increasing and customer demand keeps pushing the system performance and energy efficiency, the development of modern microprocessors has become a daunting task. The bright side is that in these commercial or non-safety-critical electronics, software and hardware bugs typically account for majority of the errors. The radiation-induced soft errors, therefore, can be tolerated much more than those in the safety-critical applications. The aggressive solutions such as TMR is prohibitively expensive, and in most circumstances, are deemed as overkill. Seeking cost-effective soft error mitigation schemes is therefore vitally important.

As discussed in previous sections, cost-effective protections for general logic are especially a major concern. There are no effective solutions, especially for complex microprocessor pipelines. The motivation of the work in this thesis is therefore to fill this gap. This work focuses on developing cost-effective soft error mitigation techniques for general logic, and providing a comprehensive SER treatment (design, evaluation model, reliability analysis) for commercial microprocessor pipelines.

The objectives of the research presented in this thesis are summarised as follows:

- Develop cost-effective soft error-mitigation techniques suitable for protecting the general logic of non-safety-critical applications. Try to overcome the significant limitations of the previous techniques by providing acceptable trade-offs between the reliability and error-tolerance overheads.
- Realise self-checking capabilities in the developed techniques. Allowing the techniques to be robust to both the errors occurring in the original circuitry and the errors occurring in the redundant parts. Minimise the overheads required by the self-checking feature to keep the technique cost-effective for non-safety-critical applications.
- Implement the developed techniques using modern CMOS technologies. Develop validation and evaluation mechanisms to comparatively evaluate the proposed techniques with previous ones. Demonstrate the advantages of the proposed techniques in terms of overhead-efficiency and error-tolerance capabilities.
- Incorporate the developed techniques into modern microprocessor pipelines, and achieve a complete and efficient pipeline protection mechanism. Validate the techniques at the system-level by modelling transient fault effects through simulations. Comparatively analyse the efficiencies and the cost of the proposed pipeline protection mechanisms with previous techniques.

1.7 Thesis Organisation

Chapter 2 provides an overview of the literature and essential background related to the work in this thesis. The advantages and potential drawbacks of the techniques proposed in the literature are also discussed, allowing clear comparisons to be carried out with the work proposed in this thesis. The traditional error coding techniques, dual modular redundancy, and triple modular redundancy techniques are introduced first. Then, the existing radiation hardened cell designs are summarised. Based on these basic ideas, different previous mechanisms for protecting the memories and general logic of microprocessors are introduced. Finally, software-level error protection techniques and some other hardening techniques are also discussed.

In completion of the research objectives introduced in Section 1.6, this thesis presents three main contributions which are described in Chapter 3, 4, and 5, respectively.

Chapter 3 presents the first contribution of the thesis by proposing two novel radiation hardening flip-flop architectures, named SETTOFF. The SETTOFF architectures can address both SEUs and the captured SETs originating from the preceding combinational gates. An evaluation model that can statistically estimate the reliability of the SETTOFF designs is also developed. The SETTOFF architectures are implemented in both 65nm and 120nm technologies and are comparatively evaluated with previous radiation hardened cell designs. The results show that SETTOFF can provide better error-tolerance capabilities with less or comparable overheads compared to most previous techniques. Therefore, it has the potential to effectively protect the pipeline of microprocessors.

Chapter 4 presents the second contribution, by proposing a self-checking technique based on the SETTOFF architectures. The self-checking technique further improves the reliability of the SETTOFF, by allowing SETTOFF to mitigate both the errors occurring in the original circuitry, and the errors occurring in the redundancies added for error-tolerance. The self-checking technique is achieved by using a self-checker, which can be shared by multiple SETTOFFs to minimise the overheads. The SETTOFF-based self-checking capability overcomes the drawback of most previous techniques which cannot address the errors occurring in their error-tolerance circuitry. In addition, the proposed technique is implemented in a register file architecture, and is compared with the widely used ECC-protected register file. The evaluation results show that the proposed self-checking technique produces significant lower soft error failure rates for both Single-Bit-Upsets and Multiple-Bit-Upsets affecting the register file. Meanwhile, the proposed technique also requires much smaller power and delay overheads, but bigger area than the ECC technique for protecting the RF.

Chapter 5 proposes the last contribution, which is a complete pipeline protection mechanism for microprocessors. The pipeline protection is achieved by incorporating the SETTOFF-based self-checking architectures into the error vulnerable sequential cells of the pipeline. The radiation-induced transient faults occurring in each stages of the pipeline are address by the radiation hardened cells incorporated in the corresponding stages. An architectural replay recovery scheme is developed to recover the relevant errors detected by the self-checking SETTOFF architecture. The proposed pipeline protection mechanism is implemented in a 32-bit OpenRISC microprocessor in 65nm technology. A gate-level transient fault injection and analysis technique is developed to analyse the reliability of the robust processor. The evaluation results show that the proposed mechanism can effectively mitigate both SEUs and SETs occurring in different parts of the pipeline. It overcomes the drawback of most previous pipeline protection techniques and achieves a complete and cost-effective pipeline protection.

Chapter 6 concludes the findings and contributions of the work in this thesis. Suggestions for future work directions are also provided.

1.8 Publications

The contributions of the work presented in this thesis have been published in the following papers:

- Y. Lin; Zwolinski, M., "SETTOFF: A fault tolerant flip-flop for building Cost-efficient Reliable Systems," On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International , vol., no., pp.7,12, 27-29 June 2012
- Y. Lin, M. Zwolinski, and B. Halak, "A Low-Cost Radiation Hardened Flip-Flop," in Design, Automation Test in Europe Conference Exhibition (DATE), 2014.
- Y. Lin and M. Zwolinski, "A Cost-Efficient Self-Checking Register Architecture of Radiation Hardened Designs," in International Symposium on Circuits and Systems (ISCAS), 2014.
- Y. Lin, M. Zwolinski, and B. Halak, "An Energy-Efficient Radiation Hardened Register File Architecture for Reliable Microprocessors," in Silicon Errors in Logic - System Effects (SELSE), 2014.
- Y. Lin, M. Zwolinski, and B. Halak, "An energy efficient radiation hardened register file architecture," in Designing with Uncertainty - Opportunities and Challenges Workshop, 17 - 19 Mar 2014. 3pp.
- Y. Lin, M. Zwolinski, and B. Halak, "Low-Cost Radiation Design: A Novel Pipeline Protection Technique for Microprocessors," in IEEE Transactions on VLSI (In preparation)

The following informal presentation has also been given:

- Y. Lin, M. Zwolinski, "A Cost-efficient Error Mitigation Technique", 6th Cisco Innovation in Test Conference (CITC), 2013.

Chapter 2

Literature Review

This chapter provides a broad overview of the state-of-the-art research that is related to the topic in this thesis. The literature relevant to each of Chapters 3, 4, and 5 is separately introduced, allowing the contributions described in these chapters to be in the context of the relevant research. Section 2.1 and Section 2.2 describe the traditional error control coding and TMR techniques, respectively. Section 2.3 summarises the existing radiation hardened cell designs, which are related to the hardened designs proposed in Chapter 3 and Chapter 4 of this thesis. The memory and cache protection techniques are summarised in Section 2.4. Section 2.5 introduces the previous microprocessor pipeline protection techniques, which are related to the proposed pipeline protection techniques described in Chapter 5. In addition, several software-based error mitigation techniques are briefly introduced in Section 2.6, while Section 2.7 includes some other error mitigation techniques. Finally, Section 2.8 summarises the literature introduced in this chapter.

2.1 Error Control Coding Techniques

This section introduces error coding techniques, which are traditional ways of providing soft error mitigation. In order to provide comparisons with the techniques proposed in this thesis, some error coding techniques are applied and comparatively evaluated in Chapter 4.

2.1.1 Parity Code

A parity code is an error-detecting coding technique which can detect (but not correct) an odd number of errors in a codeword [4]. The number of check bits in parity code is 1, regardless of the number of the data bits. There are two types of parity code, even and

odd. In an even parity code, the parity check bit is encoded such that the total number of '1s' in the entire codeword is always even. For an odd parity code, the number of '1s' in a codeword is always odd. During the decoding process, a single or an odd number of errors will change the parity of the codeword, and will therefore be detected.

2.1.2 Hamming Code

Hamming codes are a class of binary linear block code, which is typically expressed as $Hamming(n, k)$. k denotes the number of data bits, and n is the total number of bits in a codeword. For any positive integer $r \geq 2$, the following relationships apply to the Hamming codes:

- Length of codeword: $n = 2^r - 1$
- Length of data bits: $k = 2^r - r - 1$
- check bits: $r = n - k$

Hamming codes are systematic codes in which the original input data is embedded in the encoding output (i.e. the codeword). The Hamming code can detect and correct single-bit error in a codewords. The check bits in Hamming codes are arranged such that different erroneous bits trigger different error results. In a 7-bit codeword, for instance, 3 check bits are required to generate 7 different error results. The error results cannot only detect the error, but also locate the particular error bit.

The encoding and the decoding process of the Hamming code can be expressed by the generator matrix (G) and parity-check matrix (H), respectively. The parity-check matrix is constructed by Equation 2.1, where I_r is an $r \times r$ identity matrix and P is an $r \times k$ binary matrix [25]. Each column of matrix P has a Hamming weight no less than two [1]. A column in matrix P can be altered without affecting the effective codewords.

$$H = [P, I_r] \tag{2.1}$$

The generator matrix (G) can be derived from (H) using Equation 2.2, where P^T is the transpose of P and I_k is an $k \times k$ identity matrix.

$$G = [P^T, I_k] \tag{2.2}$$

[1] The Hamming weight of a codeword is the number of nonzero components in the codeword.

Let $V = (v_1, v_2, \ldots, v_n)$ be the n-bit vector representing the codeword, and the k-bit vector $D = (d_1, d_2, \ldots, d_k)$ represent the data bits. The encoding process of Hamming code can be expressed by Equation 2.3, where all additions are performed in modulo 2.

$$V = D \cdot G \tag{2.3}$$

During the decoding process, the codeword read from a computer memory, for instance, may not be the same as the original codeword written into the same memory location. Let the n-bit vector $U = (u_1, u_2, \ldots, u_n)$ be the codeword that needs to be decoded, and $E = (e_1, e_2, \ldots, e_n)$ be the error vector. Then Equation 2.4 applies:

$$U = V + E \tag{2.4}$$

An error in the i_{th} position of U is indicated by a nonzero component e_i of E. The error and its location are indicated by the syndrome vector S obtained by Equation 2.5. If all the components in S are zero, U is error-free. Otherwise S is used to determine the error location.

$$S = U \cdot H^T \tag{2.5}$$

The Hamming coding algorithm can also be explained visually by Table 2.1. The check bits are parity bits ($p1$, $p2$...) located in the position that are powers of 2 (i.e. 1, 10, 100, etc). Each parity bit covers all the bits where the bitwise AND of the bit location and the parity location is nonzero. Each bit is covered by a unique set of parity bits, such that the error location can be identified during the decoding process.

Table 2.1: Hamming code algorithm.

Bit position		1	2	3	4	5	6	7	8	9	10	11	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	
Parity bit coverage	p1	×		×		×		×		×		×	...
	p2		×	×			×	×			×	×	
	p4				×	×	×	×					
	p8								×	×	×	×	

2.1.2.1 SEC-DED Code

The error detection and correction capability of a code is decided by the minimum Hamming distance d, which is the minimum of components that two codewords differ from each other. Assume that a set of codes has a minimum distance $d = 2e + 1$. Then

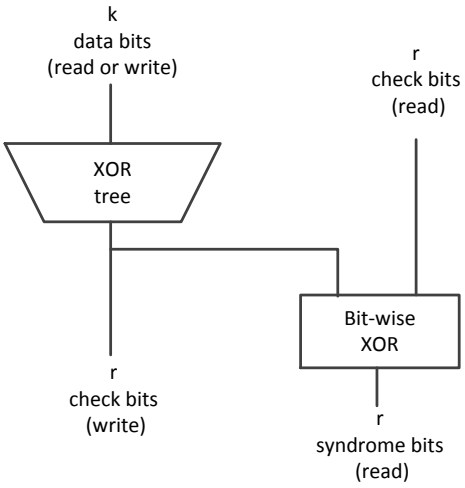

Figure 2.1: Check bits encoding and syndrome bits generation [25].

the code is capable of correcting e errors, or it can detect $2e$ errors if correction is not attempted [26]. The minimum distance of Hamming codes is 3, which means that it can only correct a 1-bit error since it cannot distinguish a double bit error in a codeword from a single bit error in another codeword.

The Single-Error-Correction Double-Error-Detection code, abbreviated as SEC-DED code, provides a remedy for the Hamming code. SEC-DED code adds an additional parity bit to extend the minimum distance to 4. By doing this, SEC-DED code can distinguish a single-bit error from a double-bit error. Therefore, it can correct single-bit errors and detect (but not correct) double-bit errors at the same time.

The SEC-DED code is a commonly used ECC technique in computer memories. The hardware implementation of encoding and decoding circuitries are shown in Figure 2.1 and Figure 2.2. Apart from this circuit, an r bit redundant register for storing the check bits, and control logic for timing are required. During write operations, check bits are generated through the encoder, which is constructed by an XOR tree according to Equation 2.3. The check bits and data bits are then stored in a particular memory location. During read operations, the syndrome bits are generated from Equation 2.5, typically by using the same XOR tree used for the encoding. The syndrome bits are then decoded by a decoder constructed by n $r-input$ AND gates to allocate the position of the error bit. The outputs of the syndrome decoder are then used to invert the code bit error through the error corrector constructed by n $2-input$ XOR gates. An OR-gate is used to generate the error signal from the syndrome bits. A NOR gate is used for signalling the uncorrectable (UE) errors, which could be the double-bit errors. Notice that the multiple-bit errors could be falsely corrected or detected by this architecture [25].

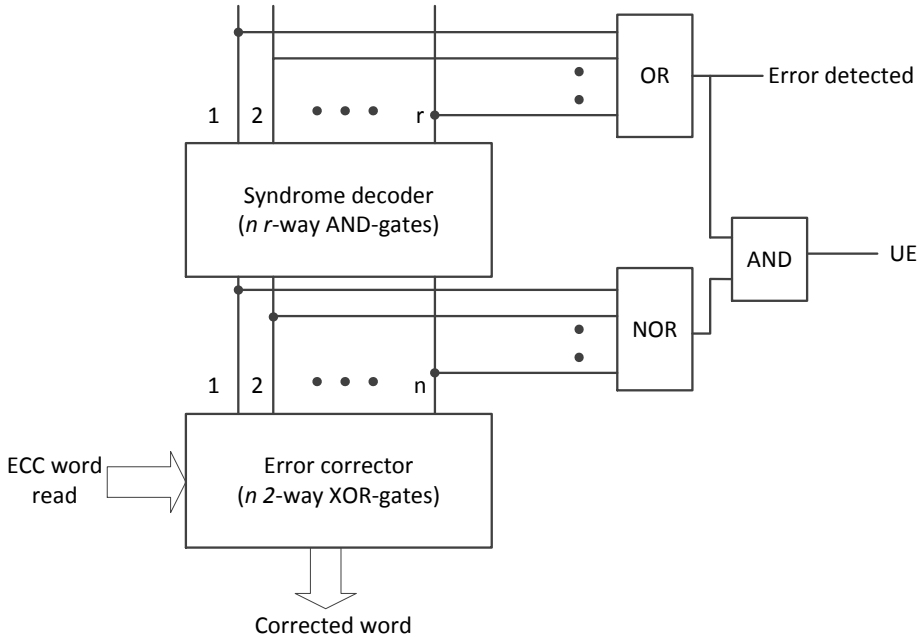

Figure 2.2: Error detection and correction circuitries [25].

2.2 Triple-Modular Redundancy and Dual-Modular Redundancy

Triple-modular Redundancy is a technique frequently used in safety-critical applications to provide high error-tolerance capabilities. TMR can be applied at different levels of a system. Figure 2.3 shows a possible TMR configuration of a system constructed by three microprocessors [27]. The outputs of the memories of all the three processors are fed back into a majority voter, which generates the inputs for each processors. Any errors occurring in one of the three copies will be outvoted by the majority voter, hence the inputs of the three processors remain error-free. TMR can also be applied at other levels, such as the gate level, where it can be used to construct TMR latches [28] [29]. The overhead of TMR in terms of power consumption and area will be over 200% more than that of the original block being protected.

The Dual-modular Redundancy (DMR) is an alternative choice which requires less overheads than the TMR [4]. Unlike TMR, DMR only duplicates the components that need to be protected. A comparator can be used to detect errors in either copy, by identifying inconsistencies at the outputs of the 2 copies. However, DMR cannot correct errors since it cannot distinguish which copy produced the correct result when an error is detected.

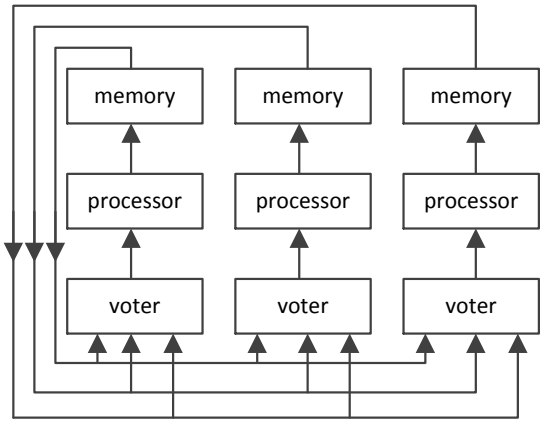

Figure 2.3: TMR configuration [27].

2.3 Radiation-Hardened Cells

This section describes previous radiation-hardened cell designs. Some of them are efficient for protecting memories, while others are suitable for protecting logic. Different techniques provide different error mitigation capabilities, but also incur various error-tolerance overheads. In general, the hardened cell designs can be divided into two categories:

1. The cells based on radiation-hardened architectures. (Section 2.3.1 to Section 2.3.9)
2. The cells based on increasing the critical charge. (Section 2.3.10)

For the cells in the first category, extra circuity is normally required to combat transient faults. The operations of such cells do not rely on specifically sizing the transistors, thus they are easily scalable. The drawback is that noticeable overheads could be incurred by the extra radiation-hardened architectures. In the second category, cells are normally hardened by increasing the node capacitance, which will increase the critical charge. Such approaches normally require specific sizing of the transistors. Therefore, the implementations of these techniques can be more complicated. It can also be more difficult to transfer such techniques into new technology generations. The advantage of the cells in the second category is that they may require less overheads compared to those in the first category.

This section summaries the error-mitigation capabilities and overheads of several existing techniques in both categories. The advantages, potential drawbacks and limitations of each techniques are also discussed.

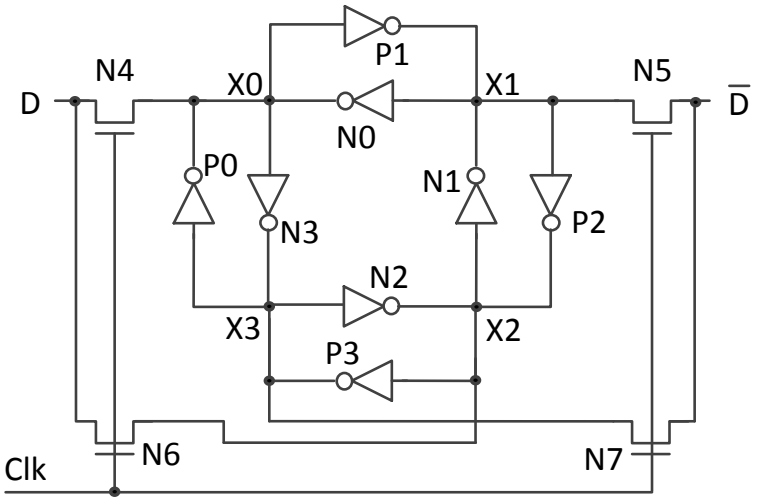

Figure 2.4: Principle of the DICE architecture [22].

2.3.1 Dual Interlocked Storage Cell (DICE)

Calin, et al. [22] proposed an SEU-immune storage cell called Dual Interlocked Storage Cell (DICE). DICE duplicates the storage element (i.e. the cross-coupled inverter pair) of a conventional storage cell to construct a so-called dual node feedback control loop. The operating principle of DICE can be explained by Figure 2.4. The four nodes $X0$ to $X3$ store the state of the cell and are accessed simultaneously through the transmission gates during write or read operations. The inverters shown in the figure are either single N-type transistors or P-type transistors according to their labels. Assume that the state stored in the cell in nodes $X0$ to $X3$ is 0101. The horizontal inverter pairs $N0$-$P1$ and $N2$-$P3$ are conducting, while the vertical ones $N1$-$P2$ and $N3$-$P0$ are blocked. This allows the state stored in nodes $X0$-$X1$ to be isolated from the same state stored in $X3$-$X2$. Similarly, when the state in $X0$ to $X3$ is 1010, the vertical invert pairs are conducting to act as normal storage elements, and the horizontal ones are blocked. The transistor-level design of the DICE memory cell is shown in Figure 2.5.

A particle strike at any node of the DICE cell can incur a transition. For example, a negative transition in a node will propagate through the P-type inverter, and induce a positive transition in one of its adjacent nodes connected to it by the blocked inverter pairs. However, a negative transition cannot affect the other adjacent node which is connected to it by the conducted inverter pairs. This is because the negative transition will be blocked by the N-type inverter. As a result, only 2 nodes can be affected by a single negative SEU. The other 2 unaffected nodes will correct such SEUs through the feedback loop. Positive SEUs are tolerated in a similar manner.

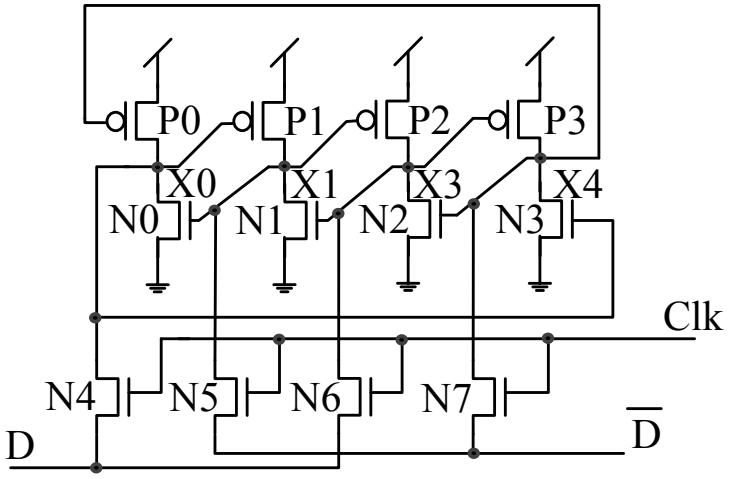

Figure 2.5: Transistor level design of a DICE memory cell [22].

DICE can be used to realise SEU mitigation in SRAM, by replacing the conventional 6-transistor SRAM cell with the 12-transistor DICE cell. The area overhead is close to 100%. Additional wires and capacitance are required by the extra write line in DICE, which will incur a performance overhead. The power dissipation would also be increased noticeably since the cross-coupled inverter pairs are duplicated. DICE can also be used for protecting sequential logic (latches and flip-flops). Rennie, et al. have implemented a radiation-hardened flip-flop based on the DICE architecture in 40nm process [30]. According to the evaluation results presented by the authors, the DICE-based flip-flop incurs a power dissipation overhead of 78%, compared to a conventional master-slave flip-flop. The delay of the DICE-protected flip-flop is also increased by 67%.

There are three major drawbacks of the DICE cell: it is not capable of addressing SET pulses occurring at the input of its write line, which may corrupt all the states stored in the 4 internal nodes. The SEU-correction process is achieved using a feedback loop and is not immediate; therefore it generates a glitch upon a correction. The glitch may propagate and corrupt the following stages when DICE is used in sequential logic. Finally, DICE is not able to mitigate MBUs induced by one particle strike at 2 internal nodes that are connected by the conducting inverter pairs. Interleaving methodologies to optimise circuit physical organisations and separate the MBU-sensitive node pairs need to be applied to increase the robustness of DICE against MBUs [31].

2.3.2 Feedback Redundant SEU/SET-Tolerant Latch (FERST)

A soft error tolerant latch architecture, namely the feedback redundant SEU/SET-tolerant latch (FERST), was proposed by Fazeli, et al [32] [33]. FERST duplicates

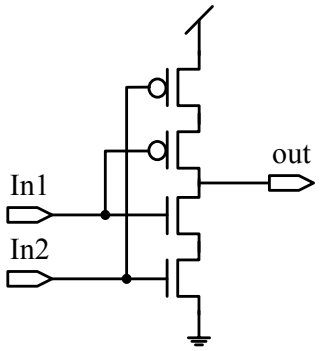

Figure 2.6: The C-element [33].

the storage element (the cross-coupled inverter pair) in a conventional latch and uses C-elements to tolerate upsets. The C-element, shown in Figure 2.6, inverts its inputs when the 2 inputs are identical (i.e. normal mode), and holds the previous state when the 2 inputs are inconsistent (i.e. the filtering mode). The FERST latch, shown in Figure 2.7, used 3 C-elements to construct a redundant feedback loop. FERST has four internal nodes, $N1$ to $N4$, which store 2 copies of the latch states. Nodes $N1$ and $N2$ are protected by C-element1 and C-element2 against SEUs. Any state changes in either nodes $N1$ or $N2$ will result in inconsistent inputs for both C-element1 and C-element2, which will then be switched into the filtering mode to prevent the upset from propagating to nodes $N3$ and $N4$. The unaffected state preserved in nodes $N3$ and $N4$ will hence correct the upset in either nodes $N1$ or $N2$ through the feedback loop. C-element3 is used for protecting nodes $N3$ and $N4$ against SEUs. SEUs occurring in either $N3$ or $N4$ will corrupt $N1$ or $N2$, respectively. Such SEUs will switch all the three C-elements into their filtering modes to block any errors propagating to the output of the latch. The output in such circumstance is in high impedance, and therefore requires a weak keeper to hold its state.

The SET-tolerant capability of FERST is realised by the delay element added to one of the two input lines of the latch. If the delay generated by the delay element is greater than the width of the SET pulse, C-element1 and C-element2 will filter the SET pulses occurring at the input. This is because the pulse and its delayed version will induce inconsistent value at the input of the C-elements. However, if the delay of the delay element is smaller than the width of the pulse, there is a possibility that the pulse will be captured by the latch.

FERST is applicable for protecting latches in logic. It can also be used to construct a master-slave flip-flop architecture, and hence can be used to protect mainstream flip-flop based design. The advantage of FERST is that it can tolerate both SEUs and SETs. However, it suffers from the following major drawbacks: Although FERST is

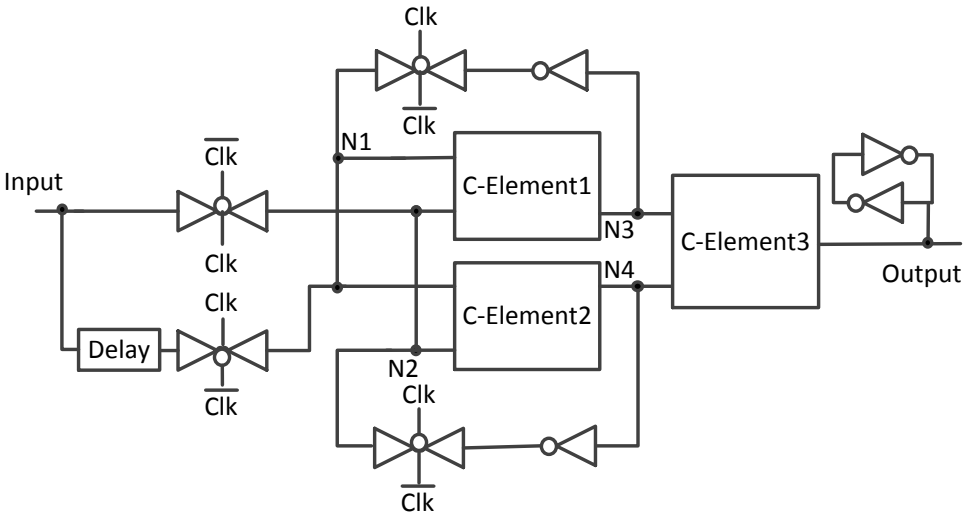

Figure 2.7: The architecture of the FERST latch [33].

cost-effective compared to a TMR latch, it still incurs around 100% area and power consumption overheads. The C-element3 added in the signal path also incurs a noticeable delay overhead of around 70% in 65nm technology. In addition, C-element3 and the keeper at the output of the latch are not protected. It has been reported by Anghel, et al. that C-elements are suspectable to single event effects and can produce substantial SERs [34]. The state stored in C-element3 can be corrupted by SEUs when it is in filtering mode, which will cause an erroneous output. C-element3 and the keeper may also generate SETs at the output of the FERST latch when C-element3 is in normal mode.

2.3.3 Built-In Soft-Error Resilience (BISER)

Mitra, et al. proposed a built-in soft-error resilience (BISER) technique [35], which re-uses the flip-flops used for scan to protect the system against particle strikes. The architecture of a BISER cell is shown in Figure 2.8. The scan portion is a redundant flip-flop added for the purpose of design-for-testability. All the scan flip-flops in the system are connected together to construct one or more shift registers. They provide high-quality testing and debugging since all the internal nodes are accessible. BISER re-uses the scan flip-flop to act as a duplicate of the system flip-flop during normal system operations (the scan flip-flop is still used for testing during test mode). The outputs of both flip-flops are connected to a C-element. An SEU that corrupts either copy of the flip-flop will be filtered out by the C-element, thus the actual output of the BISER flip-flop remains unaffected.

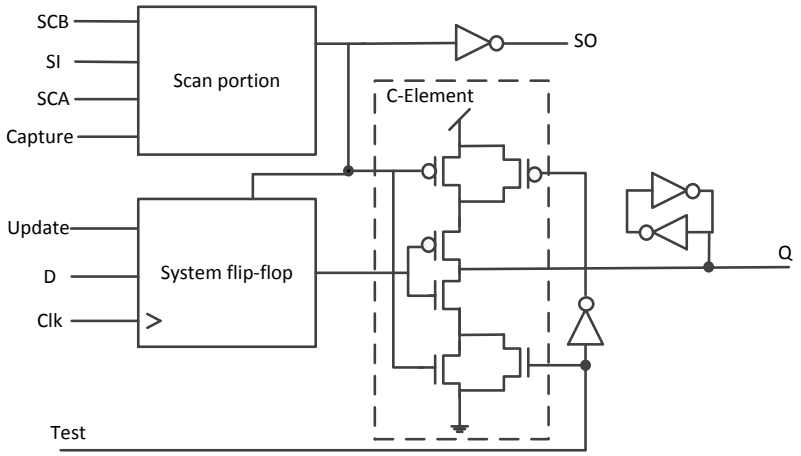

Figure 2.8: The architecture of the BISER flip-flop [35].

BISER is only applicable in design-for-test (DFT) applications with built-in scan flip-flops. The re-use of the scan flip-flop as a duplicate allows BISER to have a relatively small area overhead. However, the C-element added to the signal path will incur a noticeable delay overhead. Also, BISER cannot address SETs from the preceding combinatorial gates. In addition, as with FERST, the C-element and the keeper at the output of the cell are state-holding elements, but are not protected. Any transient faults that they produce can corrupt the output of BISER directly.

2.3.4 Time Redundancy-Based Soft Error Mitigation

Nicolaidis proposed the time redundancy-based transient error tolerant techniques, [36] [37], primarily for mitigating SETs captured by sequential logic gates. The idea is that the SETs occurring in the combinatorial logic only manifest themselves for a limited period of time, and will be recovered automatically. In other words, the presence of the correct value will still dominate in the time domain when SETs occur. Based on this observation, the time redundancy-based error-tolerance technique moves hardware redundancy (TMR or DMR) into the time-domain.

2.3.4.1 Time Redundancy-Based Error-Tolerance Architecture (TRT)

Figure 2.9 shows the Time Redundancy-based soft error-Tolerance (TRT) architecture [36]. Three latches, which are driven by three clocks each being delayed by δ, are used for sampling the output of the combinatorial block at three different time instances. The delay element δ is set by Equation 2.6, where W_{SET} is the biggest width of the SET pulse that need to be tolerated, and D_{setup} is the setup time of the latch.

$$\delta = W_{SET} - D_{setup} \tag{2.6}$$

Equation 2.6 ensures that any SET pulses with the width no greater than W_{SET} cannot be captured by more than one of the three latches. A majority voter is then used to outvote the erroneous value when an error is captured by any of the three latches. The output of the voter is hence always error-free, and will be forwarded to the actual system latch.

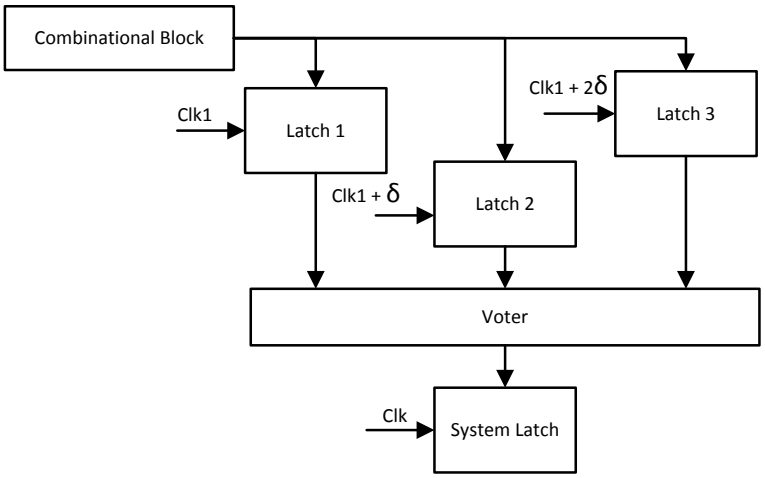

Figure 2.9: Time redundancy-based error-tolerance architecture [36].

2.3.4.2 Time Redundancy-Based Error-Detection Architecture (TRD)

In the TRT architecture, three latches are used to sample the output of the combinational block when it is stable. The hardware overheads are relatively big, and certain timing constraints are also incurred since the output of the combinational block needs to remain stable until all three latches finish sampling. In order to reduce the overheads and loosen the timing constraints, the Time Redundancy-based error Detection (TRD) architecture can be applied to only detect errors. The TRD architecture is shown in Figure 2.10 [36]. A redundant latch, Latch1, is used to sample the output of the combinational block with a delay δ, compared to the system latch. Similar to the TRT architecture, Equation 2.6 applies. The SETs with pulse widths no greater than W_{SET} can only be captured by one of the two latches. A captured SET will cause inconsistent outputs from the two latches, and will hence be detected by the comparator. Note that SEUs that occurring in either the system latch or latch1 before the comparison is conducted will also be detected.

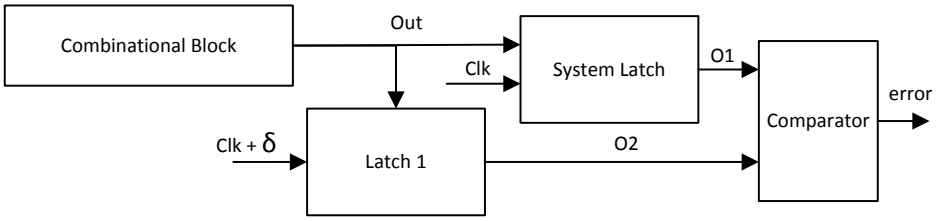

Figure 2.10: Time redundancy-based error detection architecture [36].

2.3.4.3 Cost-Reduced TRD Architecture

An optimised version of the TRD architecture was proposed, the architecture of which is shown in Figure 2.11 [37]. The new architecture noticeably reduced the error-tolerance overheads by removing the extra latch in the original TRD architecture. The comparator compares the output of the main flip-flop against its input at two time instances. This scheme maintains the same SET detection capability as the original TRD architecture (Figure 2.10). Figure 2.12 shows the operation timing diagram of the cost-reduced TRD architecture. The delay element of δ can be set by Equation 2.7.

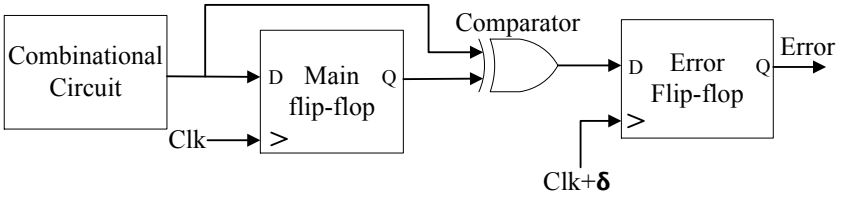

Figure 2.11: The optimized TRD architecture [37].

$$\delta = W_{SET} + D_{setup} + D_{comp} \tag{2.7}$$

where D_{comp} denotes the delay of the comparator, and D_{setup} denotes the setup time of the *Error Flip-flop*. If SETs with widths no greater than W_{SET} are captured by the main flip-flop at time $t0$, they will recover at time $t0 + \delta - D_{setup} - D_{comp}$. The comparator will then assert the error signal due to its inconsistent input values. The time between $t0$ and $t0 + \delta - D_{setup} - D_{comp}$ is referred to as the **TRD interval** in this thesis. The *Error Flip-flop* can be shared by multiple cost-reduced TRD-based flip-flops in a system. Therefore, the cost-reduced TRD technique is rather cost-effective as only a comparator is required. However, it can only detect errors, but cannot correct. The SEU-detection capability of the cost-reduced TRD architecture is also reduced compared

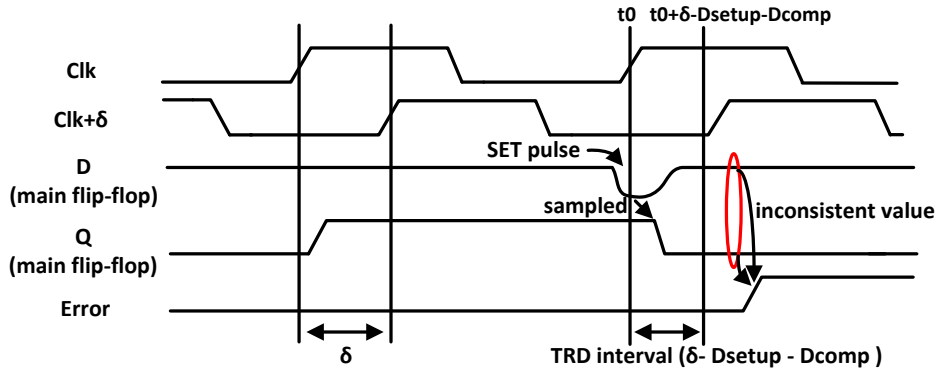

Figure 2.12: The timing diagram of the cost-reduced TRD architecture.

to that of the original TRD. The cost-reduced TRD can only detect SEUs that corrupt the main flip-flop during the TRD interval. Other SEUs occurring in the main flip-flop outside the TRD interval will escape detection. The escaped SEUs could be dangerous to the system, as unlike SETs, SEUs cannot be recovered until the flip-flop is overwritten by the next input value.

2.3.4.4 The Detection Capability of TRD

As introduced in Section 2.3.4.2 and Section 2.3.4.3, the TRD architectures can detect all the SETs manifest on the output of the combinational circuit with durations no greater than W_{SET}. They can also detect the transient pulses with durations larger than W_{SET}, but not all of them. In the original TRD architecture, a pulse will escape detection when it overlaps the latching edge of both latches. In the optimized TRD architecture, a pulse that overlaps both the sampling edge of the main flip-flop and the time when the comparison is conducted will escape. On the other hand, the SET pulses originating from the internal nodes of the preceding combinational circuit can be broadened during propagation, particularly through re-convergent paths (the multiple paths starting from one point and re-converging to another) [38]. Hence, in order to cover all the SETs that arise from the combinational circuit, the W_{SET} should be chosen as the maximum duration of the transient pulse manifest on the input of the sampling sequential gates.

2.3.4.5 Timing Constraints of the Cost-Reduced TRD Architecture

The delay element ($\delta = W_{SET} + D_{setup} + D_{comp}$) in the cost-reduced TRD architecture may incur certain performance penalties. In the case of $\delta < 0$, which means that the comparison happens before the main flip-flop samples its input, the clock period needs

to be increased by δ to ensure that the value at the output of the combinational logic is ready when the comparison is executing. The delay overhead is inevitable in this case.

In the case of $\delta > 0$, the comparison is executed after the main flip-flop finishes sampling. The output of the combination block needs to remain stable until the comparison finishes. This can be realised by utilising the the existing delay of the combinational block. If the delay of the shortest path in the combinational block D_{short} is big enough, such that the new output of the combinational block from the next cycle will not arrive until the comparison finishes, the TRD circuit can operate at the original clock frequency. However, if D_{short} is not big enough, it needs to be augmented by using slower gates or adding buffers to meet such a timing constraint.

2.3.5 Global Reliability Architecture Approach for Logic (GRAAL)

The SET-tolerant capabilities of the time redundancy-based techniques rely on the delay, δ. With a bigger δ, SETs with wider pulse widths can be covered. However, as discussed in Section 2.3.4.5, δ incurs timing constraints. Such timing constraints can be difficult or expensive to satisfy, to achieve the required error-tolerance capability. In order to conquer the limitations of the time redundancy-based technique, Nicolaidis, et al. proposed a technique called Global Reliability Architecture Approach for Logic (GRAAL) [39] [40]. GRAAL adapted the TRD architecture (Figure 2.11) by splitting the system flip-flop into a master and a slave latch. The combinational circuit at the input of the main flip-flop is also split for matching the master and slave latches. In other words, the GRAAL technique is achieved by changing the flip-flop based pipeline architecture into a latch-based architecture. In a latch-based design, when one pipeline stage is computing, its adjacent stages are in steady states and their outputs are stable. The timing constraint of the GRAAL architecture can be loose, and a conformable SET-detection interval (i.e. bigger δ) can be achieved more easily. However, GRAAL is rather complex to realise in the mainstream flip-flop based systems, because the flip-flop based architecture needs to be changed into a latch-based architecture.

2.3.6 SEM and STEM Cell

Two soft error-mitigation registers, namely, Soft Error Mitigation (SEM) and Soft and Timing Error Mitigation (STEM) techniques are proposed in [41]. Both of them applied the TRD technique in a variant TMR architecture to achieve error-mitigation for sequential logic at gate-level.

Figure 2.13 shows the gate-level embodiment of a SEM cell. Three registers R1, R2 and R3 are driven by CLK1, CLK2 and CLK3, respectively. $DataIn$ is captured by the three registers at three different time instances. Error detection is achieved by comparing

the outputs of the three registers. *Error* and *Benign* signals are generated from the comparison to invoke the recovery process. *Load_Backup* is the control signal for the recovery operation.

Table 2.2 shows the operating principle and the error-tolerance capability of the SEM cell. It can tolerant SEUs occurring in either R1, R2 or R3, and the SETs occurring in the preceding combinational logic. The recovery operation is only invoked when R1 is corrupted by soft errors (Case I of Table 2.2). During the recovery process, the values stored in R2 or R3 are loaded back into the register R1. Meanwhile, since the erroneous data in R1 has been sent to the next stage when the error is detected, a single-cycle stall is required to stall all other SEM cells, to prevent the erroneous data from contaminating the following stages. Re-computations will then conduct after the stall cycle, to overwrite the erroneous data. Since SEM does not invoke recovery operations when false errors, i.e. the errors in R2 and R3, are detected, it eliminates the unnecessary recovery overheads in the traditional error-tolerance schemes. On the other hand, SEM incurs a "small performance overhead" during normal operation, as the data in R1 can be used by the next computation stage as soon as it is ready. The detection circuitry of SEM is moved away from the signal path.

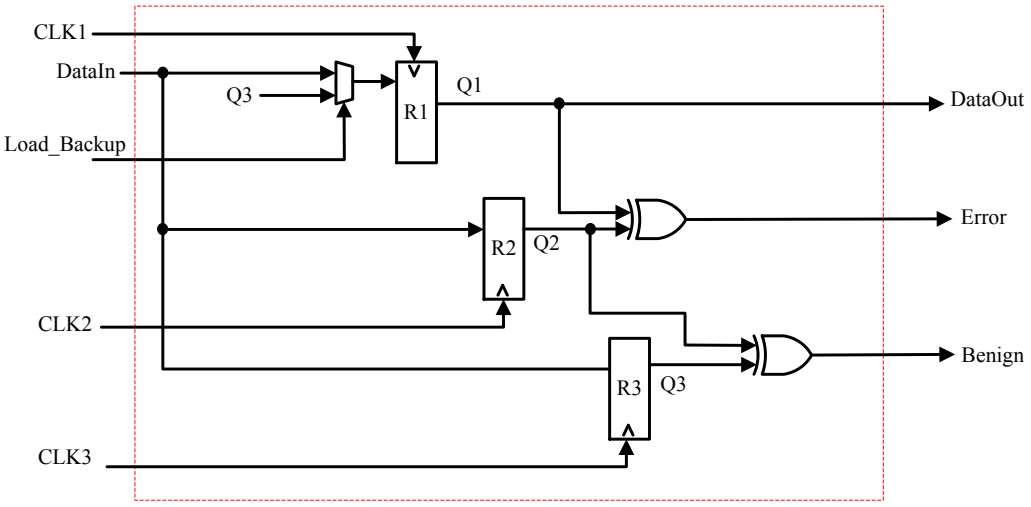

Figure 2.13: SEM cell [41].

Table 2.2: Operating principle of SEM cell (NE = No Error; SE = Soft Error; TE = Timing Error) [41].

Case	R1	R2	R3	Error	Benign	Error Recovery
I	NE	NE	NE	0	0	No recovery required
II	SE	NE	NE	1	0	Load R2 or R3 into R1
III	NE	SE	NE	1	1	No recovery required
IV	NE	NE	SE	0	1	No recovery required

Figure 2.14 shows the gate-level embodiment of the STEM cell, which can tolerate both soft errors and timing errors. Similarly to SEM, STEM uses three registers to sample the input data at three time instances. However, in STEM, the delayed clocks CLK2 and CLK3 can ensure no timing errors are sampled again by the two redundant registers.

The operating principle of the STEM cell is shown in Table 2.3. Once R2 samples the input data, it is compared with the data sampled by R1. Since R2 is timing safe, this comparison detects timing errors in R1 and soft errors in both R1 and R2. When a mismatch is indicated by the *Error* signal, R3 will be shielded from the input data, and the content in R3 will be used in the recovery process to restore R1 and R2 before the re-computation is triggered (see cases II, III, V, and VI). If no mismatch is indicated by the *Error* signal, R1 and R2 are error-free, R3 will then capture the new input data for conducting another comparison. The comparison result will be indicated by the *Panic* signal. As shown in case IV, the asserted *Panic* signal indicates that a soft error occurred in R3, which will be restored by the data in R2 during the recovery process. No re-computation is required in case IV. Since timing errors are tolerated by STEM, it was used to achieve an over-clocking system to improve system performance in [41].

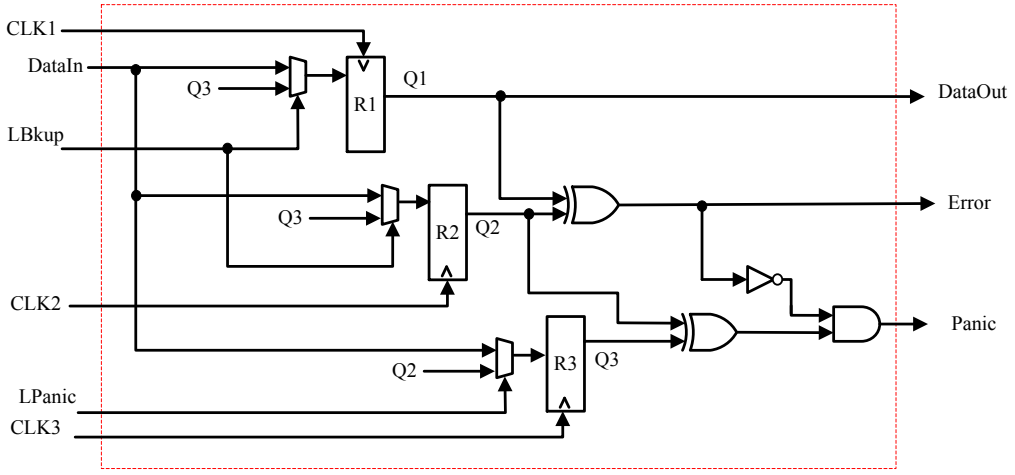

Figure 2.14: STEM cell [41].

Table 2.3: Operating principle of STEM cell (NE = No Error; SE = Soft Error; TE = Timing Error) [41].

Case	R1	R2	R3	Error	Panic	Error Recovery
I	NE	NE	NE	0	0	No recovery required
II	SE	NE	NE	1	0	Load R3 into R1, R2
III	NE	SE	NE	1	0	Load R3 into R1, R2
IV	NE	NE	SE	0	1	Load R2 into R3
V	TE	NE	NE	1	0	Load R3 into R1, R2
VI	TE	SE	NE	1	0	Load R3 into R1, R2

The TRD technique applied in the TMR architecture provides the SEM and STEM cells with strong mitigation capabilities against both SETs and SEUs. The implementation of both cells are relatively easy since the architectures are constructed at gate-level. However, the area and power dissipation overheads of both cells can be even bigger than TMR ,because additional recovery circuitry is added.

2.3.7 Razor Technique

Das, et al. proposed the Razor flip-flop for combating Timing Errors (TEs) [42] [43]. The Razor flip-flop uses an adapted version of the TRD architecture. It adds a shadow latch to sample the input of the main flip-flop, again with a delay. A sufficient delay is applied to prevent the shadow latch from re-sampling the timing error. A captured timing error is indicated by inconsistent values in the main flip-flop and the shadow latch. The error signal triggers a recovery operation, during which the data stored in the shadow latch will be trusted and used to restore the main flip-flop. Razor can be used to realise aggressive Dynamic Voltage Scaling (DVS) systems by tolerating the timing errors caused by the over-scaled supply voltage. Although the primary purpose of Razor is to achieve power efficiency, it also naturally detects the soft errors occurring in either the main flip-flop or the shadow latch. However, Razor does not consider soft errors which corrupt the shadow latch. Thus the combined recovery mechanism may actually corrupt the system by restoring an erroneous state using the shadow latch, if the shadow latch is corrupted by radiation-induced soft errors.

2.3.8 RazorII Technique

RazorII is a newer Razor technique designed with the same aim of tolerating timing errors to achieve aggressive DVS systems [44]. RazorII can also naturally detect soft errors within a flip-flop. Figure 2.15 illustrates the architecture of the RazorII flip-flop, while its operating principle is shown in Figure 2.16. The positive-edge sensitive latch is augmented with a Transition Detector (TD) which is controlled by a Detection Clock (DC) generator. The error detection is realized by detecting illegal transitions on the internal latch node, N. To prevent valid transitions from being flagged as errors, DC disables TD within the period of the CLK-to-Q delay of the latch, to allow the latch to capture its correct input state. An architectural replay recovery signal is generated by the asserted error signal.

Unlike the original Razor, if erroneous data from the preceding logic is captured by the RazorII flip-flop, the recovery mechanism re-executes the faulty operation that wrote to the flip-flop and overwrites the erroneous state. Such an operation can only recover SETs and TEs occurring in the preceding combinational gates, since they are detected during the write cycle of the flip-flop, thus the faulty operation can be easily targeted

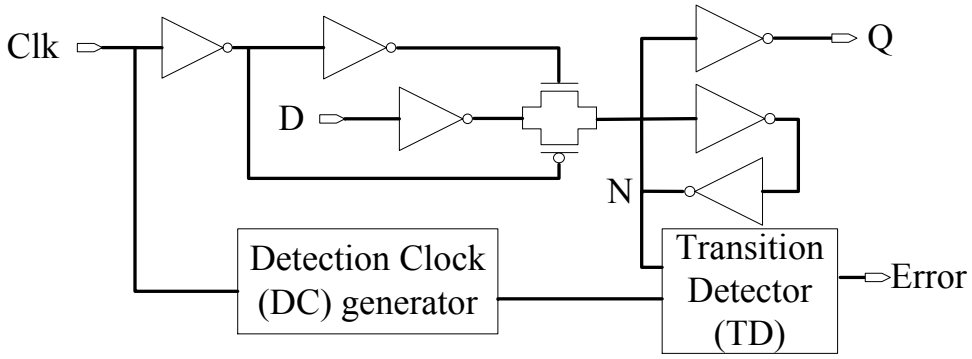

Figure 2.15: RazorII flip-flop [44].

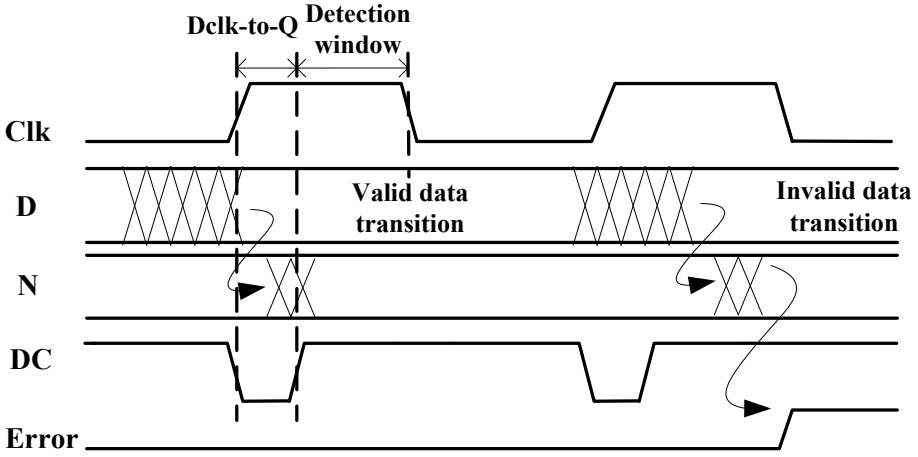

Figure 2.16: Operation timing diagram of RazorII flip-flop [44].

to re-execute. However, if an SEU is detected during a hold cycle of the flip-flop, the recovery mechanism can no longer find the latest operation that wrote to the flip-flop, hence re-executions are not achievable to overwrite the SEU. This makes RazorII difficult to use to achieve a complete error-tolerance systems.

Actually, in [44], the RazorII flip-flops are only used for protecting timing-critical pipeline registers. The architectural recovery re-fetches and re-executes the instruction in the write back stage of the pipeline, to re-write all the pipeline registers when any errors are detected. The registers storing the architectural state of the processor (such as the RF), are protected by the more expensive ECC or TMR techniques. This is because the SEUs occurring in them cannot be recovered by the RazorII recovery mechanism.

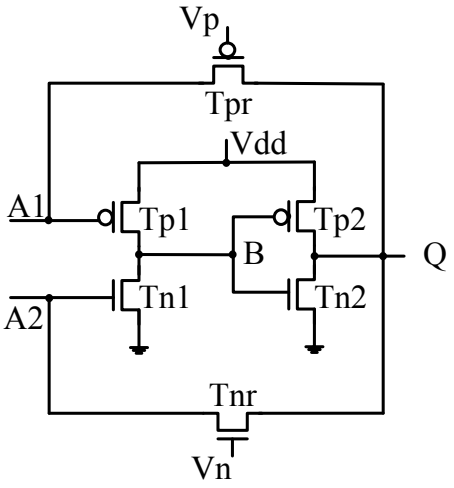

Figure 2.17: The architecture of the hardening approach using blocking feedback transistors [45].

2.3.9 Hardened Cells based on Blocking Feedback Transistors

Nicolaidis, et al. introduced a hardening approach that uses blocking feedback transistors [45]. The architecture of the hardening approach is shown in Figure 2.17. Transistors T_{nr} and T_{pr} are held open to prevent transient pulses generated at node B and Q looping back to their creating points. Meanwhile, the two nodes, $A1$ and $A2$, for the input of the first inverter are separated. Note that particle strikes on an n-transistor produce only negative current pulses, while particle strikes on a p-transistor only produce positive current pulses. Due to the separated nodes in the first inverter, the transient pulses generated by T_{nr} and T_{pr} can only turn transistor T_{n1} and T_{p1} off, but never on. Thus the transient pulses at node $A1$ and $A2$ cannot propagate to node B, but can bring node B into high impedance state. Node $A1$ and $A2$ can be restored by either the leakage current through transistors T_{nr} and T_{pr}, or by periodically causing transistors T_{nr} and T_{pr} to conduct. Either way, V_p and V_n need to be carefully adjusted.

This hardening approach can be used to construct latches or memory cells. The advantage is that it significantly reduced the area overhead compared to the DICE cell (Section 2.3.1). However, adjusting the supply voltage V_p and V_n to restore node $A1$ and $A2$ in every cell can be rather expensive and complicated, which affects the practicability of the cell. Lin, et al. in [46], introduced an 11-transistor memory cell based on an adapted version of the approach in Figure 2.17. The authors claimed that the 11-transistor memory cell can overcome the difficulties in implementing the supply voltage V_p and V_n.

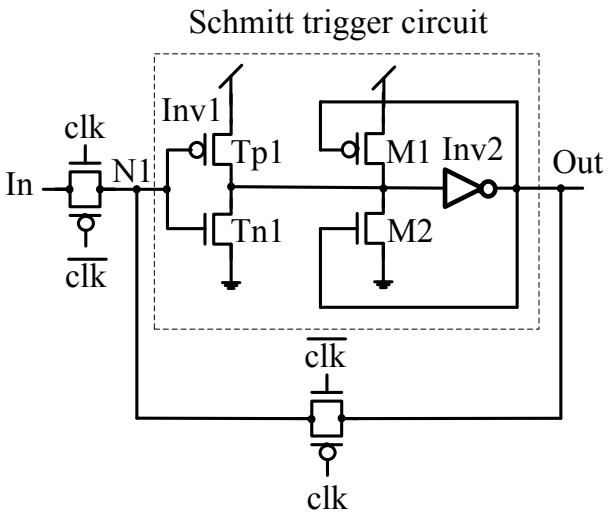

Figure 2.18: Schmitt Trigger-based soft error masking latch [47].

2.3.10 Schmitt Trigger Circuit-Based Cells

The Schmitt Trigger (ST) circuit is typically used for removing jitter in A/D converters using its hysteresis property [47]. The ST circuit has two threshold voltages, V_{th+} and V_{th-}. Rising transitions at the input of the ST circuit that exceed the V_{th+} will be interpreted as high voltage states at the output. Falling transitions that fall below V_{th-} are interpreted as low voltage states. Other signals between V_{th+} and V_{th-} do not change the output, thus the state retains. Sasaki proposed a soft error masking latch using the ST circuit to prevent SETs being captured by the latch [47]. The architecture of the latch is shown in Figure 2.18. Pass transistors are used at the input of the latch to reduce the amplitude of the SET pulses to be within the range of the two threshold voltages. The SETs with the reduced amplitudes will therefore be masked by the ST circuit shown in the dotted arc. In addition, the extra transistors, $M1$ and $M2$, operate in parallel with $Inv1$. This increases the capacitance of node $N1$, and hence increases its critical charge (Q_{crit}). The increased Q_{crit} provides node $N1$ a better immunity to particle strikes.

Lin, et al. proposed several improved versions of the ST circuit-based radiation-hardened latches [48]. The improved latches further increase the soft error immunity of the the original ST-based latch shown in Figure 2.18, by increasing the critical charge (Q_{crit}. The cascode ST latch, for instance, has 112% higher critical charge than a conventional latch [48]. Glorieus, et al. proposed an SEU-tolerant flip-flop architecture named RST (Robust Schmitt Trigger), which also relies on the ST circuit [49]. The authors demonstrated from the evaluation results that the RST cell is more power-effective than the DICE technique. Although the ST-based latches incur relative small area overheads (e.g.

12.5% area overhead for the original ST-based latch), they can only efficiently address SETs. Moreover, the ST-based techniques belong to the hardened cells in the second category. Hence they can reduce the susceptibilities of the cells to radiation strikes, but might still be vulnerable to the particles with rather high energy.

A 13-transistor memory cell, which combines the technique described in Section 2.3.9 and the ST technique, is proposed in [50]. The operating principle of the 13-transistor cell is similar to that of the 11-transistor memory cell introduced in Section 2.3.9. But the 13-transistor cell adds an ST structure to combat MBUs, by increasing the critical charge of the secondary node under particle strikes [50]. The 13-transistor cell is 33% faster than the DICE cell, but incurs 9% more area and 5% larger power penalties.

2.4 Memory and Cache Protection Techniques

As discussed in Chapter 1, Section 1.5, ECC is an effective technique for protecting memories and caches against transient faults. Apart from ECC, there are also a number of other memory- and cache-protection techniques, such as by using hardened cells. This section gives a brief overview of these techniques.

2.4.1 ECC-based Memory and Cache Protections

ECCs are widely used for protecting memories or caches, they can also be adapted to achieve more effective protections. For instance, the SEC-DED code is a commonly used ECC technique. Although SEC-DED coding can only correct single bit errors, this limitation can be addressed by applying interleaving to the memories or caches. The interleaving technique can improve the error-correction capability of SEC-DED codes, by arranging adjacent bits in a memory system to different codewords, thus decreasing the probability of MBUs occurring in the same codewords. ECCs can also be applied to achieve partial protections for memories or caches. Partially protecting only the most vulnerable bits can significantly reduce the overheads, while still being able to maintain acceptable error-tolerance capabilities. In [51], an ECC-based technique is proposed to partially protect the caches, by offering the the caches with higher access rate higher protection priorities.

2.4.2 Memory and Cache Protection Using Radiation-Hardened Cells

Using Radiation-hardened cells is an another way of protecting memories. DICE introduced in Section 2.3.1, the Schmitt trigger circuit introduced in Section 2.3.10, and the blocking feedback transistor-based technique introduced in Section 2.3.9, can all be incorporated into the memory cells to achieve radiation hardening. Radiation hardened

cells such as the RHM-12T cell (Radiation Hardened Memory cell with 12 Transistors) proposed by by Guo, et al., were demonstrated to have better SEU-immunities and comparable or lower overheads than the previous hardened cells, for protecting SRAMs [52]. The memory-protection techniques based on these radiation-hardened cells can naturally address MBUs as each individual bits are protected by hardened architectures. Such techniques can be alternative choices for applications that require high-level of MBU-tolerant capabilities.

2.4.3 Other Memory and Cache Protections

An error detection scheme called Cache Write Sure (CWS) was presented by Kim, et al., for protecting cache memories [53]. The scheme takes advantage of the pre-existing information redundancy in a multi-level caching system, and only protects the first level on-chip cache as it communicates with the CPU core most frequently.

Figure 2.19 shows the structure of the baseline design. If a request is generated for checking the data in L1 cache, the system writes the data into the Write Sure Queue. The V-unit then verifies the integrity of data by comparing the data with the pre-existing duplication in the L2 cache. The L1 cache uses a write through policy to guarantee the data coherency between L1 and L2 during the whole operation period. The buffer is used for minimising the wait time caused by the write operation in L2 cache. It is assumed that the L2 cache is error free (i.e. L2 can be protected by radiation-hardened structures), therefore it can correct the erroneous data in the L1 cache.

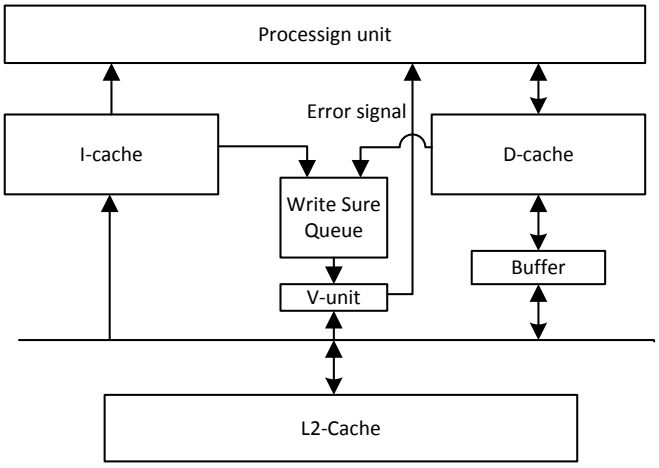

Figure 2.19: Baseline model for Cache Write Sure (CWS) system [53].

Another fault-containment scheme for cache memories is realized by re-using the redundant architectures in a TMR-protected processor [54]. It is a protocol approach that protects the cache by re-using the existing TMR architectures in the datapath.

2.5 Soft Error Protection Techniques For Microprocessor Pipelines

As discussed in Chapter 1, Section 1.5 and Section 1.6, the soft error mitigation for general logic is a much bigger challenge than for memories, and still remains open. Protecting the pipeline of microprocessors, in particular, is a major cause of concern for general logic protection. The microprocessor pipeline consists of a large number of distributed logic gates which makes it hard to effectively protect. Nevertheless, there are a number of existing soft error mitigation techniques for protecting the pipeline. Most of them focus on hardening certain sequential gates (latches or flip-flops) to address soft errors in certain parts of the pipeline. One major drawback of these techniques is that they cannot provide a complete protection of the whole pipeline efficiently against both SETs and SEUs. Some of them, for instance, are only suitable for protecting the speculative pipeline registers, while others can only protect the Register File (RF). This section summarises these existing pipeline protection techniques, and discusses their error-tolerance capabilities and limitations.

2.5.1 Pipeline Register Protection Techniques

The pipeline registers stores the instructions and intermediate data for each pipeline stage. They are typically both timing-critical and safety-critical. This is because the errors occurring in the pipeline registers can easily corrupt the control signals, and therefore result in the corruption of program executions.

2.5.1.1 RazorII pipeline protection

Das, et al. proposed a pipeline protection technique based on the RazorII flip-flop introduced in Section 2.3.8 [44]. As shown in Figure 2.20, the RazorII technique replaces all the pipeline registers that connect the combinational blocks of each pipeline stages with the RazorII flip-flops shown in Figure 2.15. This could realise timing error and soft error detection within the pipeline. Error correction is achieved by using a replay recovery mechanism at architectural level. The replay operation flushes the pipeline and re-executes the instruction in the Write Back (WB) stage of the pipeline to overwrite the erroneous state in all the pipeline registers. Such an approach can address the soft errors occurring in the combinational blocks and pipeline registers before the WB stage. This is because these blocks commit speculative executions for the pipeline, replay executions can be easily applied to overwrite all the speculative execution results. However, the soft errors occurring after the WB stage, such as the ones that corrupt the RF, cannot be addressed by RazorII flip-flops. This is because the RazorII flip-flop has a limited SEU-tolerant capability, thus cannot protect the non-speculative registers

(see Section 2.3.8). The RF in the RazorII-based design is therefore protected by a conventional ECC technique, which suffers from the common drawbacks of ECC-based RF protection techniques (See Section 2.5.2). In addition, as all the errors before the WB stage are recovered through the replay recovery process, the RazorII technique can incur large Instruction Per Cycle (IPC) overheads when the error rate is high. The IPC overhead can have an evident impact on the overall performance and energy efficiency.

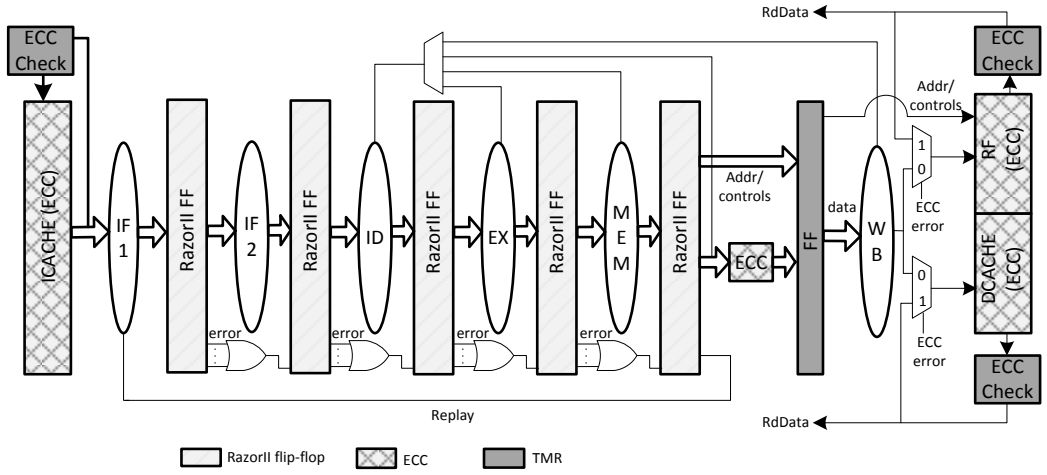

Figure 2.20: RazorII pipeline design [44].

2.5.1.2 SEM/STEM pipeline protection

Avirneni, et al. proposed two pipeline protection schemes by incorporating either the SEM or STEM cells (introduced in Section 2.3.6) into the pipeline registers of a microprocessor [41]. The pipeline design of the SEM/STEM-based techniques are similar to those of the RazorII-based techniques, since only the speculative pipeline registers are protected. However, the SEM/STEM-based techniques rely on their own error-recovery operations introduced in Section 2.3.6, rather than the pipeline replay recovery. As discussed previously, the SEM and STEM cells can provide high level of protection capabilities against both SETs and SEUs occurring in the pipeline. However, the large overheads (over 200% of area and power) incurred by the SEM/STEM cells make them unapplicable to protect big storage units such as the RF. These two techniques again, cannot achieve complete and efficient pipeline protections.

2.5.1.3 DIVA pipeline protection

Austin proposed a *Dynamic Verification Architecture* (DIVA) for protecting the pipeline with the out-of-order execution feature [55]. DIVA, which is shown in Figure 2.21,

achieves error-tolerance by adding a functional checker unit to the commit phase (CT) of a processor pipeline. The functional checker verifies the correctness of the computations from the processor core, and only permits correct computation results to pass through the commit stage (CT).

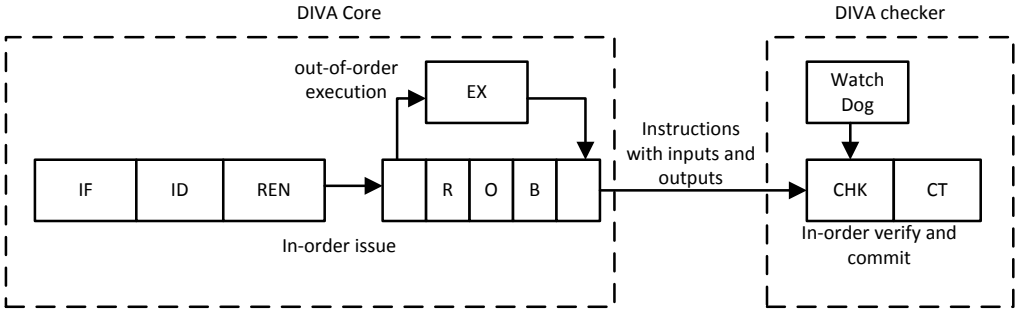

Figure 2.21: DIVA structure [55].

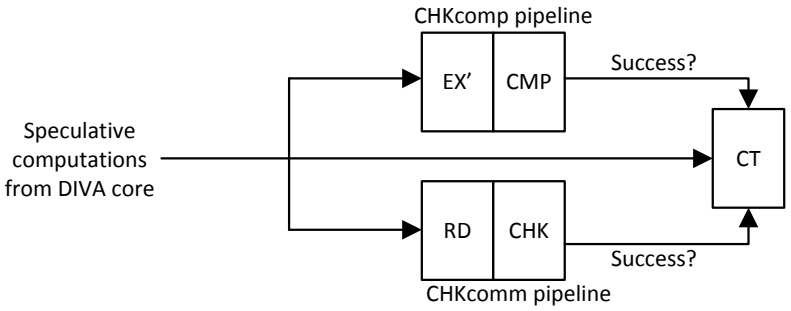

Figure 2.22: The architecture of the DIVA checker [55].

Figure 2.22 shows the architecture of the DIVA checker which consists of two pipelines, CHKcomp and CHKcomm. The CHKcomp pipeline verifies the functional units of the DIVA core. EX' is the duplicated execution block which re-executes the instructions. It can be realised by utilizing simpler algorithms to reduce the cost as it receives the instructions along with their inputs and outputs from the main core. In addition, larger transistors and timing/voltage margins are typically employed in EX' to make it robust. The CHKcomm pipeline guarantees the correct communication between combinational blocks and sequential blocks by re-executing all the communications. DIVA addresses the faults occurred in the communication process which are not considered in most conventional protection schemes. The duplicated circuitry in the DIVA checker is considered to be robust, therefore the recovery mechanism is realized by restoring the main processor core by using the duplicated data in the checker.

The drawback of the DIVA technique is that the checker re-executes all the instructions which can incur a large power dissipation overhead. The implementation of the DIVA checker is also complicated for the standard digital design process.

2.5.2 Register File Protection Techniques

The Register File (RF) in a microprocessor is the crucial part that affects the system reliability. RF stores the intermediate execution results of the processor and is frequently accessed. The errors occurring in the RF can easily propagate to other execution units and cause visible errors in the final outcome of program executions. There are generally two types of protection mechanisms for the RF in microprocessor pipelines: the ECC-based RF protections, and radiation-hardened cell-based RF protections.

Because the RF is a dense storage block, ECC can be an applicable protection technique. However, the ECCs applied in the RF are much less efficient than those are used in the memories. There are three main reasons for this: 1. The ECC bits need to be calculated and read during each operation. The performance and power overheads can be big for the frequently accessed RF, and the situation gets worse when there are multiple read ports and so multiple ECC decoding circuitries are required. 2. ECC requires a noticeably larger number of redundant bits to address MBUs. The interleaving technique introduced in Section 2.4.1 is not feasible for the RF. This is because RF is a small-size storage block, so interleaving will significantly increase the complexity of the layout, and also increase the power consumption. The chance of increasing the distance of each bit in a codeword in RF is also rather limited. 3. ECC cannot address the SETs that originate from the preceding pipeline stages and are captured by the RF. The majority of the cell area within the RF is consumed by the read and write logic. Previous research indicates that the majority of MBUs occurring in the RF are caused by captured SETs, since the combinational logic have a high degree of fanout [56]. In contrast, radiation-hardened cell-based RF protection can overcome some drawbacks of ECC-based RF protection. This section discusses previous RF-protection techniques, their advantages and limitations.

2.5.2.1 ParShield Architecture

Montesinos, et al. proposed an RF protection technique named ParShield [57]. ParShield is based on a previous published technique named Shield [58]. The Shield architecture shown in Figure 2.23 protects only a subset of the registers in the RF by generating, storing and checking the ECCs bits of those registers. It contains a table that stores ECC bits, and a set of ECC generators and checkers. These ECC-protected registers are dynamically selected during the register renaming process, and are considered as the most vulnerable registers that contain useful data.

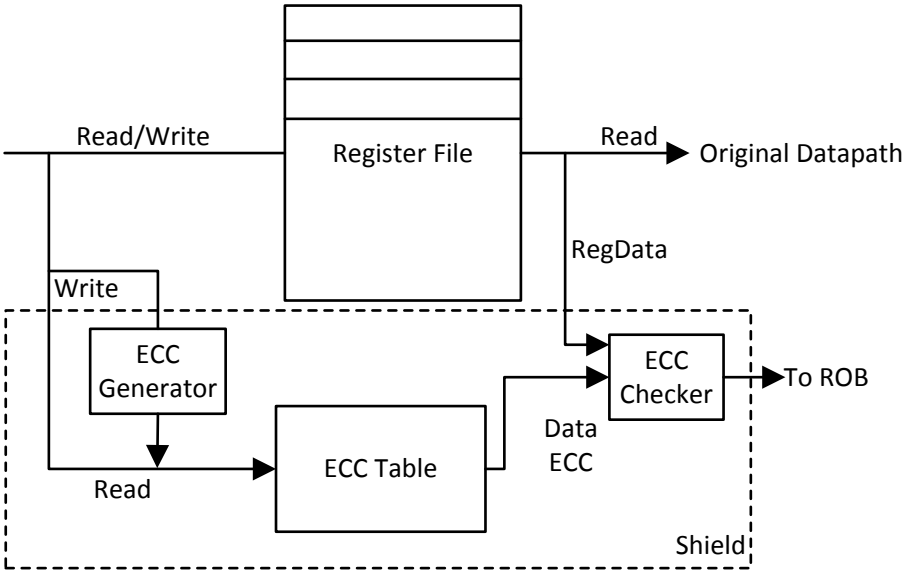

Figure 2.23: Shield architecture [57].

When a physical register needs to be written in a register renaming process, Shield generates a request for protecting the register. If the request is accepted, Shield assigns an entry in the ECC table to that register and generates the ECC bits for the data in the register. During the read process of a register that is being protected, the corresponding ECC bits will be sent to the ECC checker for verifying data integrity. When an error occurs, Shield corrects the data in the physical register on the fly. It also flushes the reorder buffer (ROB) from the latest instruction that reads the contaminated register, and then flushes the whole ECC table before resuming the system.

The selection policy of the registers to be protected is based on the lifetime of a register version. As shown in Figure 2.24, the lifetime of a physical register is defined as the period between the register allocation and deallocation during the register renaming process. One write operation and several read operations may occur during the lifetime of a register. Only transient faults occurring between the period of the write operation and the last read operation of a register can produce a visible error. This period is defined as the Architecturally Correct Execution (ACE) time [59], or the useful period of a register (Figure 2.24). The Architectural Vulnerability Factor (AVF) is defined as the fraction of time that a single bit is in its ACE state. The Shield protection scheme selects the registers that have the greatest AVF to protect. It is demonstrated in [57] that the combined useful time of all registers is small, and a small set of long-lived registers with long ACE time contribute most to the total ACE time of a system. Based on these observations, Shield primarily selects the long-lived registers which are in their ACE periods to protect. Since the ECC table is much smaller than the register file,

a replacement policy is employed when no entry is available in the ECC table for a protection request. The replacement policy either replaces an existing entry in the ECC table, or rejects the protection request by predicting and comparing the lifespan of the registers. A register with a relatively short life time will be evicted from the ECC table.

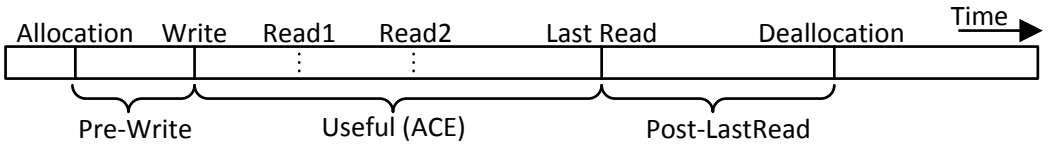

Figure 2.24: Lifetime of a register version [57].

The Parshield architecture reuses the ECC generator to add a parity bit for the entire register file to reduce the SDC AVF (Silent Data Corruption AVF) to zero. In other words, all single-bit errors will be detected by the parity.

ParShield consumes an average of 78% to 81% of the power incurred by a full ECC protection design in various applications. However, it requires 17.6% more area than the conventional ECC technique. Parshield also suffers the common drawbacks of the ECC-based RF-protection techniques.

2.5.2.2 Compiler-Guided Partial ECC RF Protection

A compiler-guided technique that partially protects the RF using ECC is proposed in [60]. This principle this technique is also based on partial ECC protection. However, this technique can be applied to the processors without dynamic register renaming support. In addition, the selection policy for the registers to be protected is based on a new concept named the Register Vulnerability Factor (RVF). RVF is used to characterise the probability that transient faults can escape from the register and thus potentially impact the system reliability. RVF is defined by the write to read and read to read intervals of a register. A register with higher RVF indicates it can potentially produce higher error rates. Based on the RVF profiling of each registers in the RF, the compiler selects the registers with the highest RVF to protect using ECC.

2.5.2.3 Duplication-Based RF Protection

Memik, et al. proposed an RF protection technique by partially duplicating the registers in the register file [61]. Since not all the physical registers are active during an execution

period of a program, the partial duplication is achieved by duplicating the active registers using the unused registers during the register renaming process. Meanwhile, parity codes are applied to identify the data integrity in the primary copy of the register. If an error occurs in the primary copy, the duplication can correct the error. Similar to the partial ECC protection techniques introduced in Section 2.5.2.1 and Section 2.5.2.2, replacement policy is essential when the register file is full, and no unused registers are available for duplications. The replacement policy used in [61] is based on capturing the register age. Among the duplicated registers, the one that has not been accessed for the longest period of time will be selected as the victim register and will be assigned to duplicate a new register. The area overhead of this technique is relatively small as it re-uses the existing redundancies in the original system.

2.5.2.4 Robust Register Cache (RRC) Technique

Fazeli, et al. proposed a partial RF protection technique named Robust Register Caching (RRC) [62]. In RRC, a small highly robust register cache memory, which is constructed using the FERST flip-flops introduced in Section 2.3.2, is added for protecting the RF. The register cache memory works with the main RF in parallel. The data stored in the most vulnerable registers in the RF are protected in the robust cache memory. Similar to the ParShield architecture described in Section 2.5.2.1, the allocation and replacement policy of the RRC protection is also developed based on minimising the AVF of a system. It is suggested that the number of read operations during a register lifetime affects the AVF of the register. The register that encounters the fewest read operations will be selected as the victim register in the replacement policy in RRC.

Since all the bits in robust cache memory are hardened by the FERST, RRC is capable of tolerating MBUs effectively. However, RRC suffers the drawbacks of the FERST flip-flop discussed in Section 2.3.2. For instance, the C-element added in the signal path in FERST can incur a large performance overhead, and the SEUs corrupting the C-element may corrupt the output of FERST.

2.6 Software-Based Soft Error Mitigation Techniques

As introduced in Chapter 1, Section 1.5, there are techniques that work at the software-level to address soft errors occurring in the hardware. From the device level to the software level, the existing number of transient faults in a system decreases since many of them are masked during the propagation through each level. Therefore, some software-based approach at the higher level of the design may need to deal with fewer errors than hardware-based approach, by ignoring the errors that are masked.

A time-redundancy based software approach is introduced in [23]. Error detection is achieved by running a duplicated program in parallel with the original program, and comparing the execution results of the two programs. This approach does not require hardware redundancies. However, the execution of the redundant program and the comparison consume 10% to 30% more time than executing a single version of the original program. Moreover, the duplicated program will incur a noticeable power consumption overhead.

Another software-based approach proposed in [63] aims to reduce the soft error vulnerability of a system rather than detecting and correcting errors. This technique reduces the time that the instructions sitting in the vulnerable storage structures, by filling the instruction queue with invalid entries rather than valid instructions during lengthy stalls. The technique reduces the probability that the valid instructions are attacked by radiation particles. Since most bits in an invalid instruction will not be read, the possibility of a fault in an invalid instruction to produce a visible error is small. The error rate can hence be reduced. The drawback of this technique is that it only address the soft errors caused by instructions, but cannot reduce the probability of the data being corrupted by radiation strikes. This is because identifying active data and the non-active data is relatively difficult.

In addition, a fault signalling technique is also introduced in [63]. The technique avoids the so-called false detected faults (the faults that are masked and do not induce visible errors) being signaled by the conventional fault-detection architectures. The technique is achieved by tracking the propagation of the faults. An error signal is only asserted when a visible error is induced by the fault during the tracking process. The fault signalling technique avoids the unnecessary correction process invoked by the false detected faults.

Yan, et al. proposed a compiler-based approach that can increase the reliability of a register file in an unprotected system [60]. The idea is based on reducing the RVF rate (see Section 2.5.2.2) by re-scheduling the instructions. The re-scheduling method aims to schedule the write operations as late as possible and the read operations as early as possible to reduce the Write - Read and Read - Read intervals of a register. The Superblock scheduling algorithm [64], hyperblock scheduling algorithms [65] and scheduling slacks [66] are employed to achieve the compiler-based re-scheduling process.

The software-based error mitigation techniques can assist the hardware-based error-tolerance architectures to further reduce the SER of a system. However, the software approaches are typically too difficult or expensive to achieve an efficient error mitigation on their own.

2.7 Other Soft Error Mitigation Techniques

This section introduces some other soft error mitigation techniques that are not categorised in the previous sections.

2.7.1 Removing the Speed Penalty of ECC Technique

The speed penalty incurred when calculating and reading the ECC bits is the main drawback of the ECC technique. Two schemes are introduced in [67] and [68], which can eliminate the delays caused by computing and reading the ECC bits. Figure 2.25 and Figure 2.26 show the diagram of the two techniques, respectively.

Figure 2.25 shows the technique that eliminates the delay penalty caused by the ECC-encoding process. The data bits and the ECC bits are stored in two separate memories. The data is written into the data memory as soon as they are ready and the calculation of the ECC bits is done in parallel. The delay caused by calculating the ECC bits in the data write path is hence eliminated. The ECC bits are checked and the errors are detected in XOR1, the correction process is conducted by the 1*hot* gate and the $XOR2$ gate afterwards.

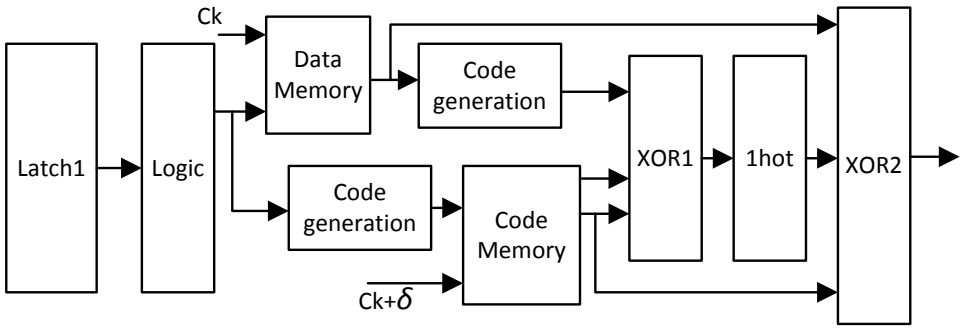

Figure 2.25: Elimination of extra delay in the write path [67].

Figure 2.26 shows the technique that eliminates the delay penalty caused by the ECC-decoding process. The error detection and correction blocks in Figure 2.25 are amended to work in parallel with the data read operations. The read operation directly provides data from the ECC protected memory to the system through a MUX. If an error is detected at some time after the read operation, all the latches will be held except latch1. This is because latch1 has already been contaminated by the erroneous data and needs to be restored. MUX then provides the correct data from the correction block to correct latch1 before the system resumes.

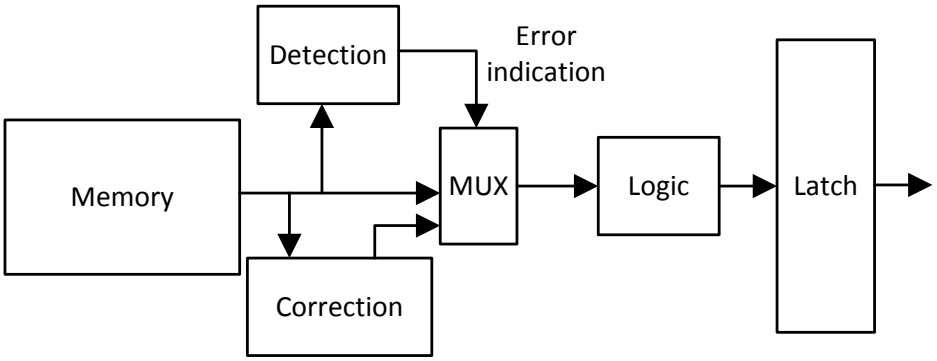

Figure 2.26: Elimination of detection and correction delays [67].

2.7.2 The Self-checking Controller with Datapath Interactions

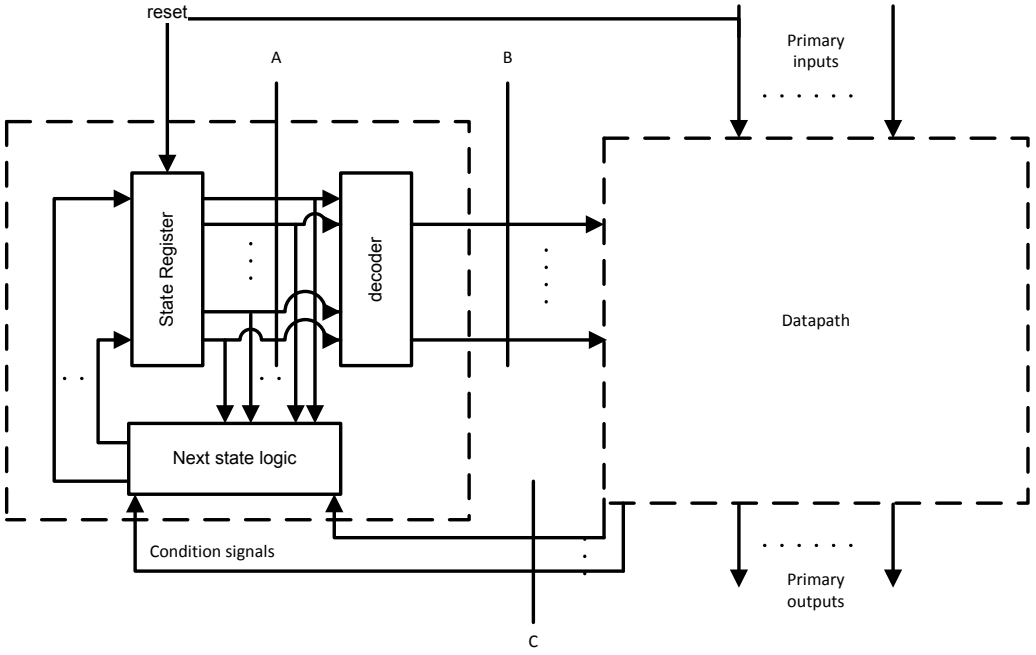

Figure 2.27: Basic controller/datapath architecture [69].

Oikonomakos, et al. proposed a self-checking system shown in Figure 2.27, which re-uses the fault-tolerant architectures existed in the datapath to protect the controller in a controller/datapath architecture [69]. Assuming that the datapath is already protected by certain fault-tolerant architectures, the errors occurring in the control signals at point B can also be detected by the existing fault tolerant architecture in the datapath. The

re-use of the fault-tolerant architecture requires only a little more cost for protecting both the controller and the datapath than for protecting the datapath alone.

2.8 Summary

The literature reviewed in this chapter suggests that traditional radiation-hardened techniques used in safe-critical applications are far too expensive for un-safety-critical commercial products. The general logic, such as the pipeline of microprocessors, are especially hard to protect. A number of techniques were proposed to combat radiation-induced soft errors in logic. However, most of them can either provide limited level of reliability, or incur large implementation overheads for commodity electronics. In addition, they lack of self-checking capability has caused the previous techniques to be still vulnerable to the soft errors occurring in the redundant circuitry added for error-tolerance. The limitations of the existing techniques make them incapable of providing a complete and effective protection for complex microprocessor pipelines. Therefore, cost-effective radiation-hardening solutions for general logic still remain open, and have become the major challenge for modern commodity microprocessor designs.

Chapter 3

Soft Error and Timing Error Tolerant Flip-Flops

It has been discussed in Chapter 1, Section 1.5 and Section 1.6 that transient faults occurring in general logic are a particular cause of concern. Using radiation hardened cells to protect the sequential elements can potentially be an effective solution for general logic. However, as summarised in Chapter 2, Section 2.3 and Section 2.5, the radiation hardened cells in the literature have various drawbacks. Most of them either provide limited error-tolerance capabilities, or incur unacceptable overheads for commercial applications. Consequently, they are difficult to use for achieving a complete and efficient protection for microprocessor pipelines. This chapter proposes a novel radiation hardened flip-flop architecture named SETTOFF. Two versions, SETTOFF1 and SETTOFF2, are developed, which can overcome the drawbacks of the previous hardened cells, and potentially provide a cost-effective protection for general logic. A reliability evaluation model is also developed which can quantify the reliability of different designs, and thus provide straightforward comparisons for the error-tolerance capabilities. Both SETTOFFs are evaluated in 120nm and 65nm technologies, and are compared with the previous techniques in terms of reliability and error-tolerance overheads.

In order to describe the error-tolerance capability of SETTOFF, Section 3.1 introduces the soft error vulnerability of a conventional flip-flop. The principle of the SETTOFF architecture is introduced in Section 3.2. Section 3.3 and Section 3.4 describe the designs of the two versions of SETTOFF in detail, respectively. The implementation issues of SETTOFF are discussed in Section 3.5. Section 3.6 proposes the reliability evaluation model. The experimental work and the comparative evaluation results are described in Section 3.7. Finally, Section 3.8 concludes the chapter.

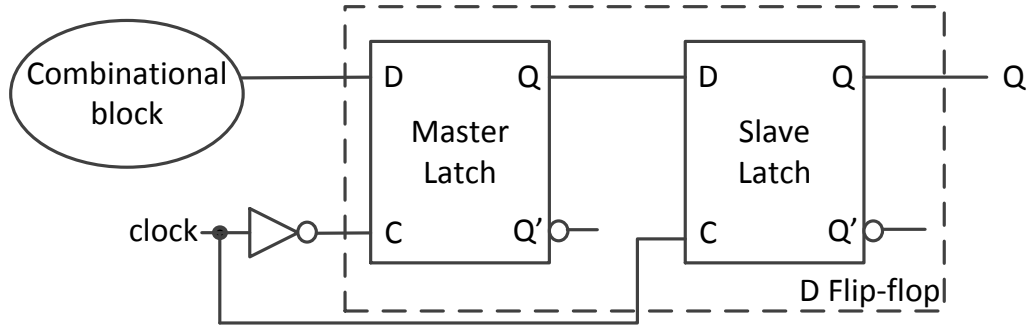

Figure 3.1: The conventional master-slave flip-flop.

3.1 Soft Error Vulnerability of the Conventional Flip-Flop

To explain the proposed SETTOFF architectures, the soft error vulnerability of the conventional flip-flop will be introduced first. Figure 3.1 shows a D-type master-slave flip-flop triggered by the positive clock edge. Radiation particle strikes can either induce SEUs that corrupt the states stored in the two latches when they are opaque, or induce SETs in the combinational circuits which can be captured by the two latches. However, only the soft errors that corrupt the output Q of the flip-flop can propagate to the following stages of the system.

During the low phase of the clock, the master latch is transparent and the slave latch is in opaque mode. The flip-flop in such a case is vulnerable to the SEUs that corrupt the slave latch, which can directly produce an erroneous output at Q. During the high phase of the clock, the master latch switches to opaque mode and the slave latch becomes transparent. In this case, the flip-flop is vulnerable to SEUs that corrupt the master latch, which will propagate through the transparent slave latch and manifest erroneous values at Q. SETs originating in the preceding combinational logic and presented at the input of the flip-flop may be captured by the master latch on the positive clock edge. The captured SETs in the master latch will again be captured by the slave latch on the negative edge of the clock, therefore will incur an erroneous Q across the entire clock cycle.

3.2 Soft Error and Timing Error Tolerant Flip-Flops (SETTOFF)

In order to combat both the SEUs that corrupt the state of the flip-flop, and the SETs that are captured by the flip-flops, we propose two versions of radiation hardened flip-flops, namely the Soft Error and Timing error Tolerant Flip-Flops (SETTOFF). The primary goal of SETTOFF is to mitigate soft errors, but as the name suggested, timing

errors occurred in the combinational logic are also naturally addressed. SETTOFF aims to provide such error-mitigation capability with minimum overheads for a microprocessor pipeline. As discussed in Chapter 1 and Chapter 2, achieving error detection and correction normally requires much bigger overheads than only achieving error-detection. SETTOFF can both detect and correct, but it minimises the overheads by conducting all the error detections at circuit-level, but separating part of the error-correction operations into the architectural-level of the processor. The rationale of this is explained as follows:

The errors that corrupt the output of a flip-flop during a write cycle are easy to recover by an architectural pipeline replay operation. This is because the instruction that caused the erroneous write operation can be easily targeted when such errors are detected. The pipeline only needs to re-execute the faulty instruction and the instructions following it to overwrite the corrupted output of the flip-flop before it contaminates the following stages. The cost for incorporating the circuit architecture for the replay operation is low since the replay is a conventional technique, which normally already exists in modern microprocessors to support speculative operations such as the branch prediction [44].

The errors which may corrupt the state stored in a flip-flop during a hold cycle are difficult otherwise to recover using the low-cost replay mechanism. This is because the replay mechanism recovers errors by re-execution and re-writing the faulty flip-flops, but when an error occurs in a hold cycle of a flip-flop, the latest instruction that writes to the flip-flop may have already retired. Therefore, the simple replay mechanism cannot target the replay point for re-writing the corrupted flip-flop. Checkpointing and rollback operations may be required for architecturally recovering the errors occurring during the hold cycle of the flip-flop [70]. However, the checkpointing and rollback mechanism is much more expensive than the replay, as it requires the system states to be check-pointed in the memory intermediately. A large power consumption overhead can be incurred due to the extra communications with the memory. The rollback also incurs a noticeable Instruction Per Cycle (IPC) overhead.

SETTOFF detects both the errors occurring during the write cycle and hold cycle of a flip-flop. The errors occurring during the write cycle will trigger a replay recovery operation at the architectural level; while the errors that occur during the hold cycle will be corrected on the fly by the built-in circuit-level architecture. The details of both versions of SETTOFF (SETTOFF1 and SETTOFF2) will be described in the following sections.

3.3 SETTOFF1

Figure 3.2 shows the architecture of SETTOFF1. The main flip-flop is a conventional master-slave flip-flop introduced in Section 3.1. For clarity, only the last state-holding

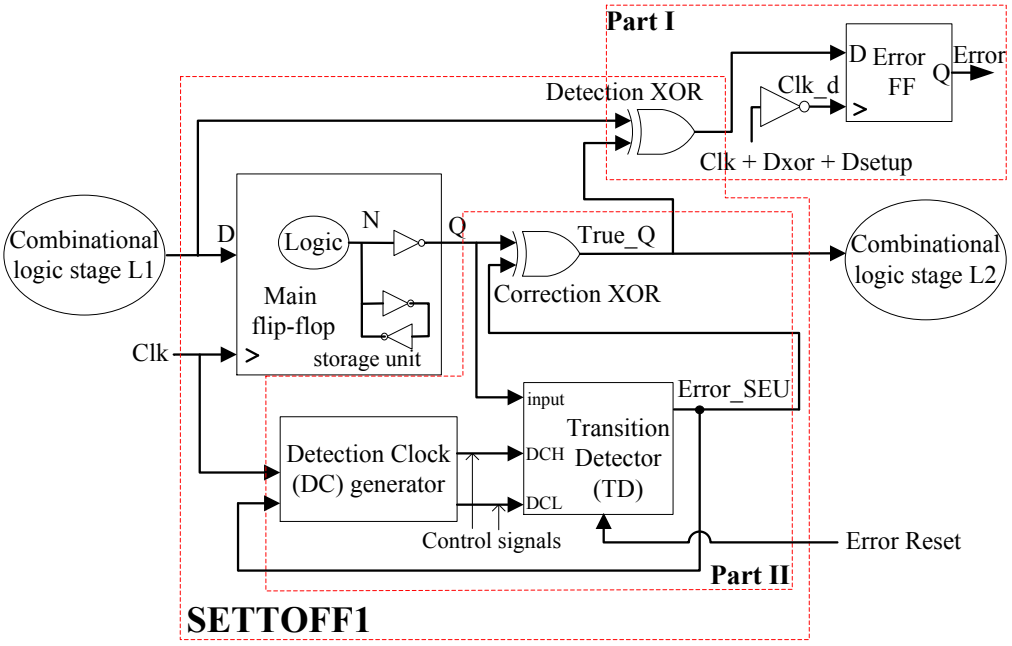

Figure 3.2: The architecture of SETTOFF1.

element (the inverter pair) in the slave latch is explicitly shown. The internal node N indicates the state held by the inverter pair. The output Q is the inverted version of N. The *logic* block precedes the cross-coupled inverter pair represents the rest of the flip-flop circuitry including the master latch and the rest of the slave latch architecture. The error-tolerance circuitry is divided into two parts:

PartI is adapted from the Time Redundancy-based Detection (TRD) architecture introduced in Chapter 2, Section 2.3.4.2. It contains a detection XOR-gate that compares the input and output ($True\_Q$) of SETTOFF1, and an error flip-flop which is driven by a delayed clock (Clk_d). The delay element (δ) is obtained from Equation 3.1.

$$\delta = D_{hclk} + D_{xor} + D_{setup} \tag{3.1}$$

where D_{hclk} is the period of the high clock phase, D_{xor} is the delay of the detection XOR-gate, and D_{setup} is the setup time of the error flip-flop. The TRD interval, which is equal to the high phase of the clock (D_{hclk}), can hence be derived from Equation 3.2. The error flip-flop is enabled during the write cycle of the main flip-flop to capture error signals. When incorporating SETTOFF1 into a register architecture, the error flip-flop can be shared by multiple bits to minimise the area and power overheads.

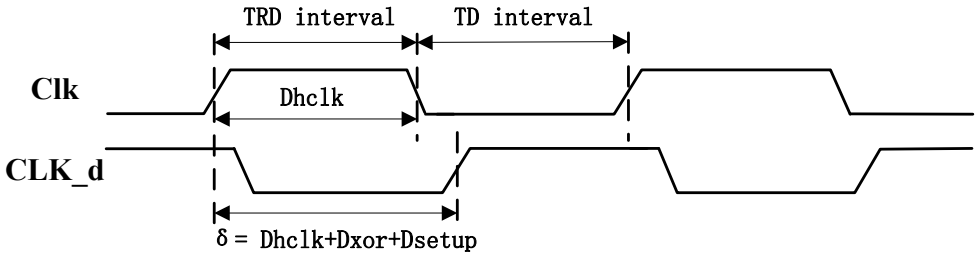

Figure 3.3: The TRD and TD interval of SETTOFF1.

$$\begin{aligned} T_{TRD} &= \delta - D_{xor} - D_{setup} \\ &= D_{hclk} \end{aligned} \tag{3.2}$$

PartII is a Transition Detector (TD) based architecture. It comprises a transition detector monitoring the output (Q) of the main flip-flop, a detection clock (DC) generator controlling the TD, and a correction XOR-gate which propagates or inverts node Q to node $True\_Q$ according to the output ($Error\_SEU$) of the TD.

3.3.1 Operating Principle of SETTOFF1

Figure 3.3 illustrates the operating principle of SETTOFF1 during a clock cycle. The clock cycle is separated into two intervals during which the two parts of the error-tolerance circuitry work in turn. PartI of SETTOFF1 works during the TRD interval which is equal to the high clock phase; While PartII of SETTOFF1 works during the TD interval which is equal to the low clock phase.

Part I is responsible for detecting three types of error occurring during the TRD interval in a write cycle of the main flip-flop:

1. As shown in Figure 3.4, the captured SETs manifest on the output of combinational logic stage L1, with a pulse width no greater than D_{hclk}, will recover before the comparison recording point. So when comparison is conducted at the comparison recording point, the captured SET presented at the output (Q) of SETTOFF1 and the recovered SET manifest at the input (D) of SETTOFF1 will cause inconsistencies at the inputs of the detection XOR-gate. Therefore these SETs will be detected. The error signal will then be latched in the error flip-flop soon after the negative edge of the clock (i.e. the comparison recording point).

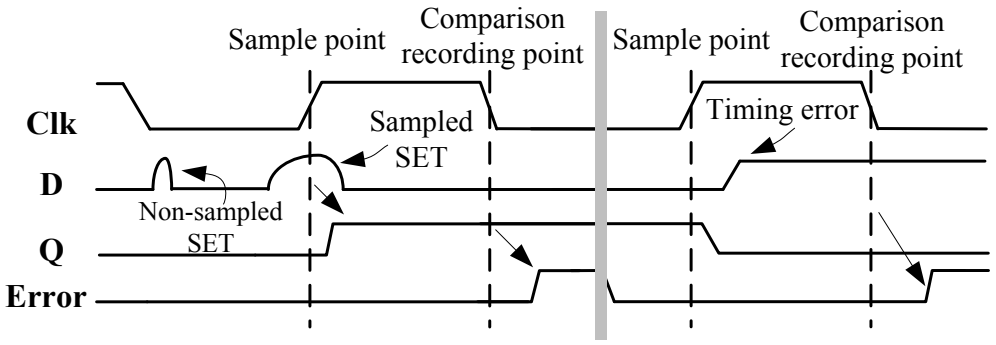

Figure 3.4: The operating principle of Part I in SETTOFF1.

2. The SEUs that flip the state stored in the master latch of the main flip-flop during D_{hclk} will propagate to the output of SETTOFF1. Similar to the SETs, these SEUs will also be detected since they will cause inconsistencies at the input and output of SETTOFF1.

3. A timing error, which is a delay error originated by the preceding combinational logic, may cause the flip-flop to fail to capture the correct input for the current cycle. As shown in Figure 3.4, the timing errors with a delay no more than D_{hclk} will be detected due to the inconsistent inputs of the detection XOR-gate.

It should be noted that the correct functioning of the TRD architecture requires the input of the flip-flop to remain stable during the TRD interval (refer to Chapter 2, Section 2.3.4.2 for details). This induces certain timing constraints during the design process. The details of the constraints will be introduced later in Section 3.5. The errors detected by the TRD architecture are the errors occurring during the write cycle of the flip-flop. The error signals are recorded in the error flip-flop, which will then trigger the architectural replay operations to recover these detected errors. The replay recovery mechanism will be explicitly described in the system-level design in Chapter 5, Section 5.3.1.

The TD-based architecture in Part II is responsible for detecting and on-line correcting the SEUs that corrupts the slave latch during the TD interval (i.e. the negative clock phase). Because these SEUs can occur during the hold cycle of a flip-flop, it is more efficient to correct them at the circuit-level. With a small overhead to the detection architecture, the TD-based architecture provides on-the-fly recovery for the SEUs detected during the TD interval.

The TD-based architecture in PartII has two operation states which are defined in Table 3.1. In the normal operation state, the output of the TD ($Error\_SEU$) is set to zero to allow node Q propagating to node $True\_Q$. The DC generator disables TD

Table 3.1: The operation states of Part II in SETTOFF1

State	Activities
Normal Operation (Error_SEU=0)	Propagate Q Enable TD during TD interval
Fault Operation (Error_SEU=1)	Invert Q Enable TD during both intervals

during the TRD interval to avoid legal transitions being flagged as errors. During the TD interval, the TD is enabled by the DC generator, and will detect any SEUs that reverse the state of the flip-flop at node Q as illegal transitions. Upon a detection, TD asserts the $Error\_SEU$ signal, which feeds into the correction XOR-gate and the DC generator.

Therefore when Q is flipped by any SEUs in the slave latch, the correction XOR-gate and the asserted $Error\_SEU$ signal will then invert Q back to the correct state at nearly the same time. This ensures the output of SETTOFF1 ($True\_Q$) to be error-free during the TD interval. Meanwhile, with the asserted $Error\_SEU$ signal, the DC generator generates control signals to enable TD during the entire clock phase (both TRD and TD intervals), such that SETTOFF1 is switched to the fault operation state.

SETTOFF1 will not return to the normal operation state until TD is reset to zero by the $ErrorReset$ signal, or TD detects the next transition in Q, which will also switch $Error\_SEU$ back to zero. To explain this, we assume two circumstances as shown in Figure 3.5(a) and Figure 3.5(b):

(a) Figure 3.5(a) shows the first circumstance in which the next transition in node Q is caused by another SEU flipping the slave latch when SETTOFF1 is in the fault operation state. The second SEU will reverse the state of the main flip-flop back to the correct value. TD detects such a transition and switches $Error\_SEU$ back to zero, which will switch the DC generator and SETTOFF1 back to the normal operation state to propagate Q to $True\_Q$. In other words, an even number of SEUs during one cycle will correct the state of the flip-flop.

(b) As shown in Figure 3.5(b), if the next transition is caused by SETTOFF1 sampling the next input, the former detected SEU will be overwritten. Since TD is also enabled during the high clock phase in the fault operation state, it will capture such transitions and reset $Error\_SEU$. SETTOFF1 is therefore switched back to the normal operation state.

A third circumstance can happen when a new input is captured by the flip-flop during the following cycle, but the new input does not reverse the corrupted state of the main flip in the former cycle. As is shown in Figure 3.5(c), in this case, the former detected

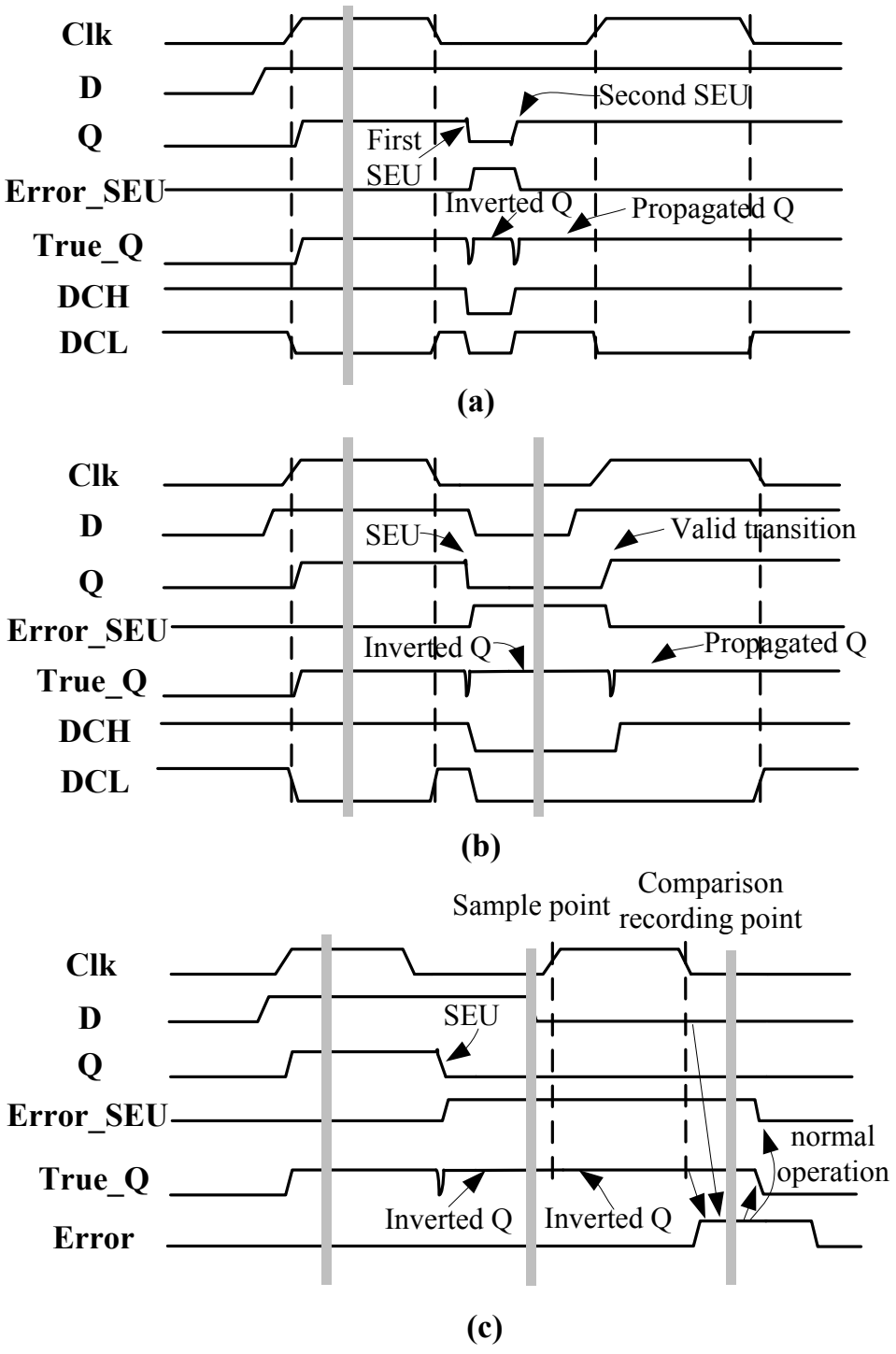

Figure 3.5: The operating principle of Part II in SETTOFF1.

SEU is overwritten, but no transition occurs in Q. SETTOFF1 therefore would stay in the fault operating state which inverts the correct Q to generate an erroneous output, $True\_Q$. However, the erroneous $True\_Q$ will not be read by the following stages since it will be detected by the TRD architecture during the TRD interval in the same cycle. The TRD architecture will then generate a replay signal to recover the erroneous output ($True\_Q$), and reset the $Error\_SEU$ through the $ErrorReset$ signal. SETTOFF1 will be switched back to normal operation after the replay recovery. For summary, Part II guarantees the output of the SETTOFF1 will never be corrupted by SEUs occurring during the negative clock phase. Other faults are detected by Part I.

Upon corrections of the SEUs occurring during the TD interval, a correction glitch (see Figure 3.5) may occur at the output of SETTOFF1 due to the propagation delay of the TD and the correction XOR-gate. Such glitches may propagate. However, if sampled, they will be detected as SET pulses in the following stage and will not corrupt the system.

3.3.2 Transistor Level Design of SETTOFF1

The circuit schematics of the TD and the DC generator are shown in Figure 3.6. The TD shown in Figure 3.6(a), is developed from the transition detector proposed for RazorII [44]. The TD requires two pulse generators to generate an 'implicit' pulse out of a rising and a falling transition at its input node, respectively. Each pulse generator is constructed by a delay-chain formed by an inverter and a transmission gate, to capture the correspondence transitions. The two delay chains are connected by four evaluation trees each constructed by two transistors connected in serial. The four evaluation trees are constructed as part of two dynamic OR-gates. When any transitions occurring at the $Input$ node, the 'implicit' pulse generated by the correspondence delay chain will conduct one of the four evaluation trees, which will either charge or discharge node M to switch the $Error\_SEU$ pin. The two control signals, DCH and DCL are generated from the DC generator, and are the on/off switches for the dynamic OR-gates on the upper and lower sides, respectively. The two sides work in turn in the two operation states described in Table 3.1.

The $Error\ Reset$ signal pre-charges node M to set the $Error\_SEU$ pin to zero and the circuit to the normal operation state. During the normal operation state, the upper side OR-gate is switched off by DCH; DCL turns on the lower side OR-gate during the negative clock phase. In this case, the inverter $I3$ and the transmission gate $TG2$ act as the pulse generator for the rising transition. The inverter $I2$ and the transmission gate $TG1$ act as the pulse generator for the falling transition. If a rising transition is captured by $I3$ and $TG2$, the generated 'implicit' pulse will momentarily conduct the evaluation tree constructed by $d1$ and $d3$ in the lower side OR-gate to discharge node M. Similarly, a falling transition will produce an 'implicit' pulse to discharge node M

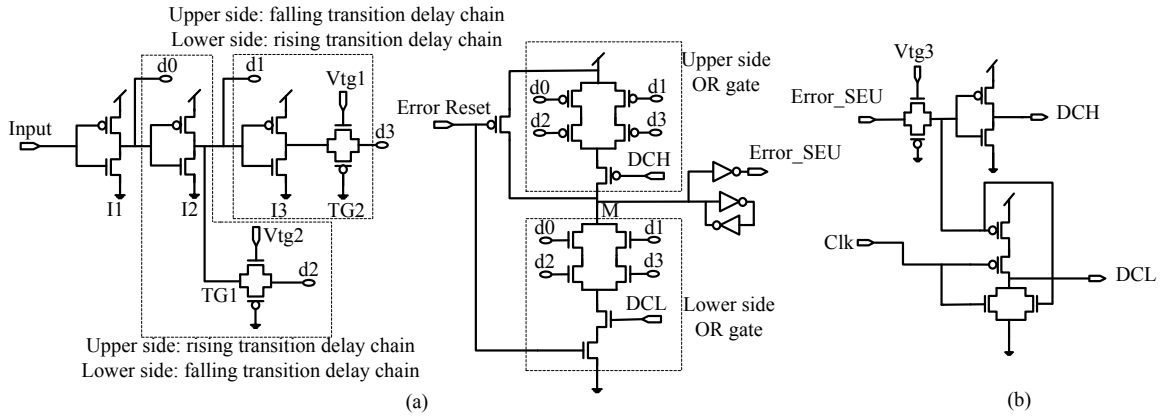

Figure 3.6: The circuit schematic of SETTOFF. (a) The transition detector. (b) The detection clock generator.

through $d0$ and $d2$. When M is discharged, $Error\_SEU$ will switch the DC generator and SETTOFF1 into the fault operation state.

In the fault operation state, the DC generator will generate a low DCL signal to switch off the lower side OR-gate. The upper side OR-gate will be switched on by DCH during all clock phases. This time $I3$ and $TG2$ are the delay chain for the falling transitions, while $I2$ and $TG1$ form the delay chain for the rising transitions. Any transitions at the input of the TD during fault operation state will charge node M through a conducted evaluation tree on upper side, and will switch SETTOFF1 back into the normal operation state.

The operating principle of DCH and DCL is given in Figure 3.5(a) and Figure 3.5(b). The $Error\_SEU$ pin will be switched between '0' and '1' in the presence of multiple transitions at the input of the transition detector.

The *Error Reset* signal can be generated in an architectural replay process invoked by the *Error* signal from the TRD architecture. The cross-coupled inverter pairs are used to protect the dynamic node N from discharging or charging by the leakage current.

The circuit architecture of the detection clock generator is shown in Figure 3.6(b). A delay is created by the transmission gate for generating DCL and DCH signal. This allows sufficient time for the dynamic node M to be charged or discharged when a transition is captured by the delay chains. The voltage supply for all the transmission gates in SETTOFF1 is tunable for controlling the delay they generate. However, the normal supply voltage (1.2V) is used when validating the design.

3.4 SETTOFF2

SETTOFF1 provides a high level of reliability by mitigating both SETs and SEUs occurring during the entire clock cycle. In the meantime, timing errors are also naturally addressed. The error-tolerance overheads of SETTOFF1 are minimised by separating the error recovery process into two different levels. However, there are some issues for SETTOFF1. The primary one is that the correction XOR-gate added in the signal path will introduce a Clock-to-Q delay overhead to the main flip-flop. In addition, the speed of the on-the-fly correction process for the SEUs detected by the TD depends on the propagation delay of the TD and the correction XOR-gate. Such a process can be slow due to the complexity of the TD and DC generator in SETTOFF1, thus a wide correction glitch can be produced upon the corrections.

In this section, we propose a second version of SETTOFF: SETTOFF2, which overcomes the weakness of SETTOFF1, and further reduces the error-tolerance overheads. The three major improvements of the SETTOFF2 are:

1) The TD-based error-tolerance architecture is significantly simplified, so that the power and area are noticeably reduced.

2) SETTOFF2 removes the Clock-to-Q delay overhead caused by the SEU-correction circuitry added in the signal path.

3) SETTOFF2 also eliminates the requirement of the TRD architecture to protect the TD-based architecture of SETTOFF1 in a complementary manner (see the third circumstance shown in Figure 3.5(c) in Section 3.3.1).

3.4.1 Operating Principle of SETTOFF2

Figure 3.7 shows the architecture of SETTOFF2. The TRD circuitry in Part I and its operating principle stay the same as that in SETTOFF1. The Transition Detector (TD) in Part II is re-designed and re-located to monitor the internal node N of the flip-flop. The DC generator is completely removed. The correction XOR gate is moved into the main flip-flop and replaces the inverter to drive the output Q. During normal operation, the output of TD ($ERROR\_SEU\_bar$) stays high, such that the correction XOR-gate acts as a normal inverter to invert node N to Q.

Similar to SETTOFF1, Part I and Part II in SETTOFF2 work in turn in the two intervals shown in Figure 3.3. During the TRD interval (i.e. when the clock is high), TD is disabled by the high clock signal and its output ($ERROR\_SEU\_bar$) stays high indicating no errors. During the TD interval, TD is enabled by the low clock signal, such that any SEUs that flip the state stored in the slave latch will be detected as illegal

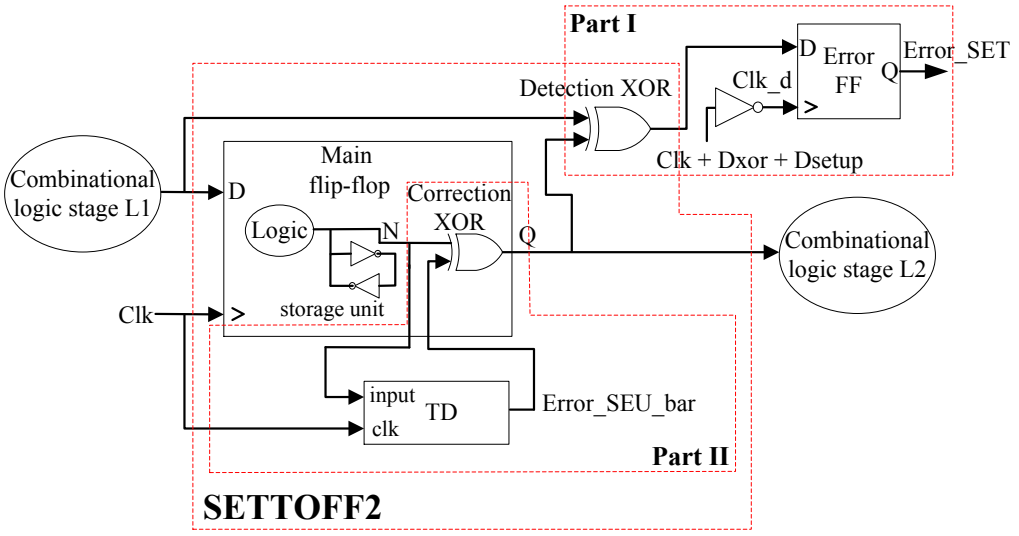

Figure 3.7: The architecture of SETTOFF2.

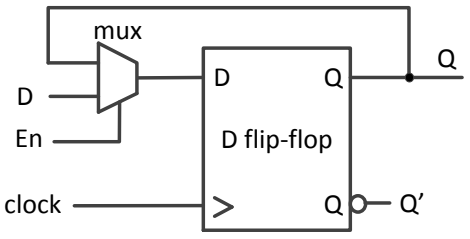

Figure 3.8: The multiplexer-based flip-flop hold architecture.

transitions at node N. *ERROR_SEU_bar* will then be assigned to zero upon detection, so that the correction XOR-gate will propagate N to correct the SEU at Q, on the fly. TD will be reset and *ERROR_SEU_bar* will be set to 1 by the next high clock signal to switch the flip-flop back to normal operation. To further illustrate this, we assume the three circumstances shown in Figure 3.10(a), Figure 3.10(b), and Figure 3.10(c):

(a) When the cycle following an SEU-correction cycle is a write cycle, the flip-flop captures a new input value so that the bit-flip error in the slave latch presented at N is overwritten at the rising clock edge. Meanwhile, the rising clock edge will also assert the *ERROR_SEU_bar* signal to let the correction XOR-gate invert N to Q as a normal inverter.

(b) When the cycle following an SEU-correction cycle is a hold cycle, flip-flops are typically held by using one of two architectures: the multiplexer-based architecture shown in Figure 3.8, or the clock-gating-based architecture shown in Figure 3.9. In the multiplexer-based architecture, the flip-flop still captures the input in a hold cycle, but

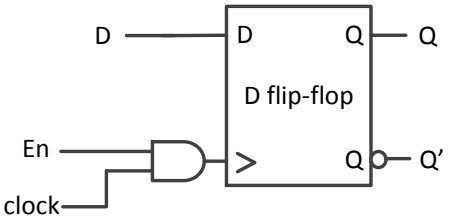

Figure 3.9: The clock gating-based flip-flop hold architecture.

the output Q is selected by a multiplexer to feed back into the input D. The flip-flop therefore captures the Q corrected in the former cycle to overwrite the SEU stored in slave latch during the hold cycle. Similar to case (a), the $ERROR\_SEU\_bar$ is set to 1 by the rising clock edge, such that SETTOFF2 is switched back into normal operation.

(c) During the hold cycle of a clock-gating-based architecture shown in Figure 3.9, the enable signal En is set to zero, such that clock driving the flip-flop is gated by the AND-gate. No input is captured by the flip-flop, hence the SEU remains in the slave latch during the hold cycle. However, the clock feeding into the TD is also gated, therefore $ERROR\_SEU\_bar$ remains at 0 to ensure that the bit-flip error stays corrected at Q.

The on-the-fly correction of the SEU in SETTOFF2 also produces a correction glitch, which again can be addressed by the SETTOFFs in the following stage of propagates. Notice that although both SETTOFF1 and SETTOFF2 are developed based-on a master-slave D-type flip-flop, they can also be adapted to protect other types of D-type flip-flops. This is because the SETTOFF architecture mitigates the errors that corrupt the last storage element of the flip-flop to ensure an error-free output. Other errors that do not affects the output of the flip-flop are masked.

3.4.2 Transistor Level Design of SETTOFF2

Figure 3.11 shows the transistor level design of the optimised TD in SETTOFF2. The construction of the delay chain is not changed from the TD used in SETTOFF1. The upper side dynamic OR-gate in the original TD is removed. The high clock signal is used to disable the TD and assert the $ERROR\_SEU\_bar$ signal. The lower side dynamic OR-gate is enabled only during the negative phase of the clock to capture transitions that will assign $ERROR\_SEU\_bar$ to zero. It should be noted that the optimised TD in SETTOFF2 significantly reduced the area and power consumption compared to the TD in SETTOFF1. The removal of the DC generator and the simplified architecture also increased the speed of the on-line SEU-correction process, which indicates a smaller SEU-correction glitch will be induced. The widths of the correction glitch of both SETTOFF1 and SETTOFF2 will be evaluated in Section 3.7.5.3.

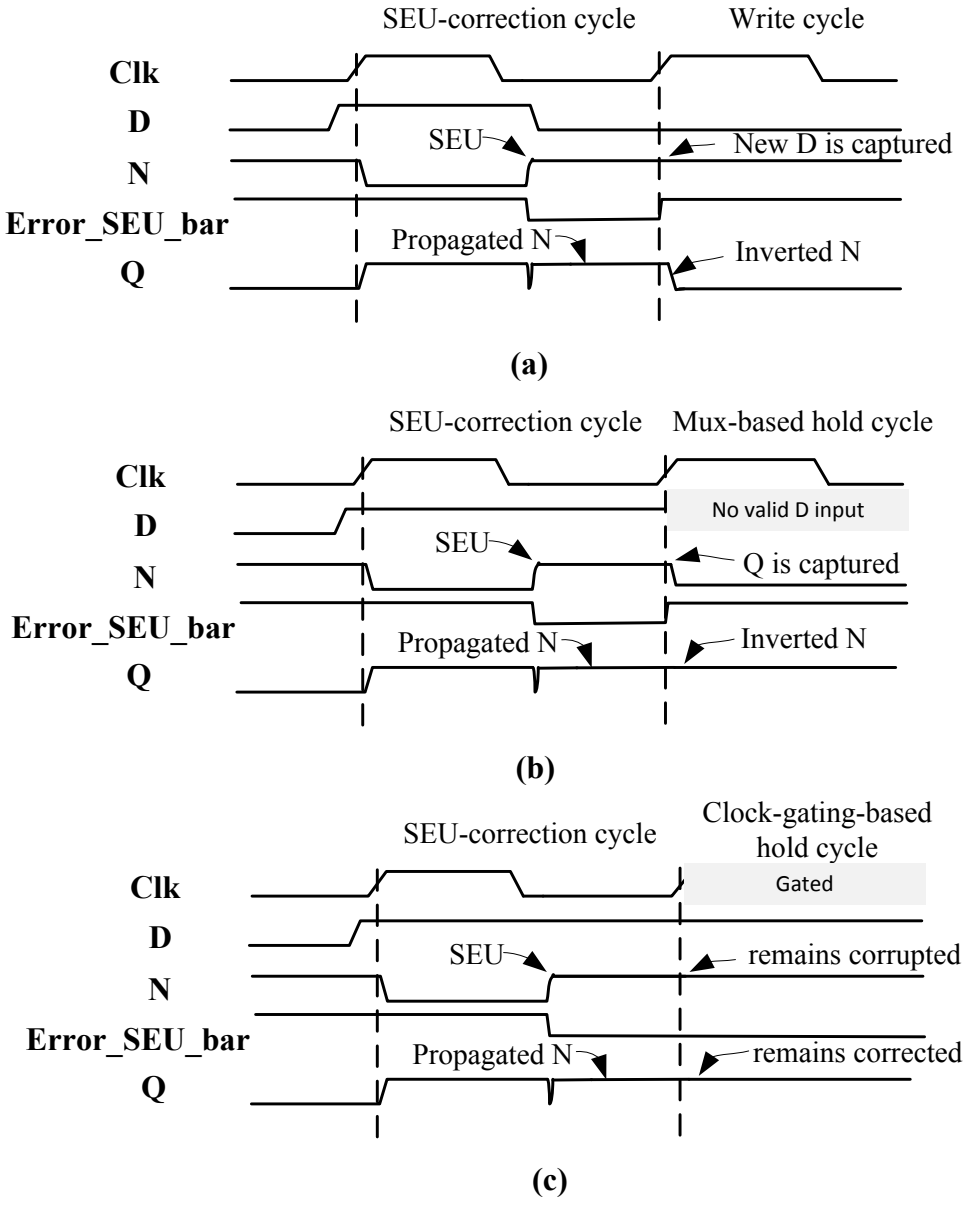

Figure 3.10: The operating principle of Part II in SETTOFF2.

The trade-off for this reduction in resource usage is that the optimised TD can no longer detect multiple SEUs occurring in the slave latch during one detection interval. This is because that the *ERROR_SEU_bar* cannot be switched between 1 and 0 by multiple transitions at the input of TD. Therefore, the design of the optimised TD is based on the assumption that the probability of a single circuit node to be struck by multiple SEUs during a single clock cycle is small, so such situations should have negligible contributions to the overall SER of the flip-flop.

Figure 3.12 shows the circuit schematic of the correction XOR-gate, which is used to replace the inverter in a conventional flip-flop to drive the output Q. The input *Error_SEU_bar* is connected to the output of the TD, which is 1 in normal operation. Therefore the Transmission Gate (TG) is blocked, and the delay of the XOR-gate equals that of inverter $I1$, which has the same drive strength and the same delay as the replaced inverter in normal operation. In other words, the SETTOFF2 architecture completely removes the extra delay path added for the SEU-correction in SETTOFF1. The increase of the Clock-to-Q delay in SETTOFF2 is only caused by the extra load added at the output of the main flip-flop.

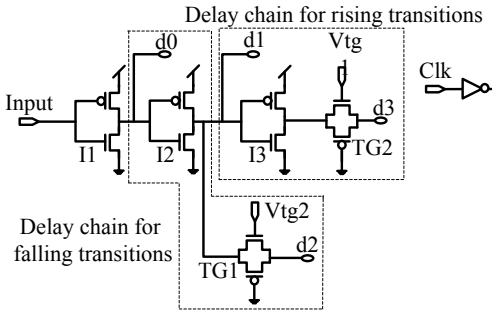

Figure 3.11: The circuit schematic of TD in SETTOFF2.

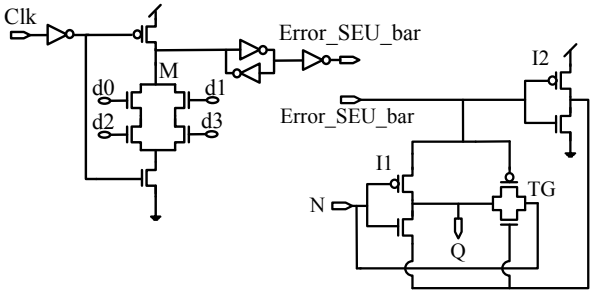

Figure 3.12: The correction XOR gate.

3.5 Circuit Implementation Issues for SETTOFF1 and SETTOFF2

Using the system clock to drive both the main flip-flop (on the rising edge) and the error flip-flop (on the falling edge) is significantly simpler than using two separate clocks. The TRD and TD intervals in SETTOFF can be altered by tuning the duty cycle of the clock without affecting the operating speed. The ratio of the widths of the two intervals represents a trade-off between the SET-detection capability and SEU-correction capability. Specifically, a wider TRD interval detects SETs with greater pulse widths and TEs with greater delays, while a wider TD interval detects and recovers more SEUs

on the fly. An asymmetric clock may be required to meet the reliability requirement in certain circumstances. However, the falling edge of the clock is not timing-critical in normal operation. The duty-cycle clock jitter at the falling clock edge only affects the TRD interval, and hence only affects the error-tolerance capabilities rather than the system operation. Clock jitter at the rising clock edge affects the system the same as with a conventional symmetric clock.

The TRD interval formed by the adapted TRD architecture incur certain timing constraints, which will be explained by the following equations:

$$T_{TRD} = \delta - D_{comp} - D_{setup} \tag{3.3}$$

$$T_{TRD} \geq W_{SET} \tag{3.4}$$

$$T_{TRD} \geq W_{glitch} \tag{3.5}$$

$$\delta \geq D_{Clk-to-Q} + D_{comp} + D_{setup} \tag{3.6}$$

$$D_{short} + D_{comp} \geq \delta \tag{3.7}$$

where:

- δ = Phase shift between Clk and Clk_d
- T_{TRD} = the TRD interval
- W_{SET} =The maximum pulse width of the SETs,
  or the maximum delay of the TEs that can be detected
- W_{glitch} = The width of the correction glitch
- D_{short} = The shortest path of the combinational logic preceding SETTOFF
- D_{comp} = Delay of the comparator in the TRD architecture
- D_{setup} = Setup time of the error flip-flop
- $D_{Clk-to-Q}$ = Clk-to-Q delay of the main flip-flop

The T_{TRD} is determined by Equation 3.3. Equation 3.4 indicates the detectable SETs and TEs, while Equation 3.5 ensures that the correction glitches generated from Part II of SETTOFF can be detected by the TRD architecture in the following stage. Equation 3.6 guarantees that the output of the main flip-flop is ready and stable before the comparison is conducted in the TRD architecture. Equation 3.7 shows the shortest path constraint of the combinational logic that precedes the TRD architecture. The shortest path constraint ensures that the output of the combination logic is not updated by the

new input value from the following cycle, until the comparison result is captured by the error flip-flop. After setting T_{TRD} according to Equation 3.3, Equation 3.4, Equation 3.5 and Equation 3.6, buffers may need to be inserted to extend the shortest path to satisfy Equation 3.7.

3.6 Statistical Analysis of Soft Error Failure Rates of the Flip-Flop

In a digital system, an SEU or an SET induced by a radiation particle strike needs to corrupt the output of a flip-flop in order to cause a visible soft error at the system level. An SEU occurring in the master or slave latch of a conventional D-type flip-flop, for instance, will corrupt the output of the flip-flop with 100% probability. An SET however, can only corrupt the output of a flip-flop if it is captured. In this thesis, the corruption of the output of a flip-flop due to either an SEU or a captured SET is defined as a 'failure' of the flip-flop. A failure of a flip-flop may or may not propagate to a higher level of the system to cause a soft error. The probability of a failure causing an error at the system output depends on the system architecture and execution. However, the overall SER of a system is the function of the failure rates of all the incorporated flip-flops, or other sequential logic. Therefore, reducing the failure rate of the flip-flop in the presents of SETs and SEUs, is an efficient way of reducing the overall SER of a system.

This section proposes an analysis model that can quantify the failure rate of various types of flip-flops caused by SETs and SEUs. The failure rate analysis model takes the variability of the circuit into account. We introduce two metrics, the SET failure rate and the SEU failure rate, which represent the failures caused by SETs or SEUs, respectively.

3.6.1 SET Failure Rate Evaluation Model

An SET generated from a combinational circuit node is a glitch with a peak that exceeds the threshold voltage. It has two important characteristics: the peak of the glitch (Vp), and the width of the glitch (w), as illustrated in Figure 3.13.

Consider a synchronous pipeline as shown in Figure 3.14. Assume an SET pulse has been generated within the Combinational Logic (CL) and has propagated to DFF2. The pulse can cause a functional failure if it is erroneously sampled (yet meets the setup and hold times of DFF2), or if it forces the device to go into the metastability state (violates the setup and hold time of DFF2). The occurrence time of the glitch, and its width determine whether or not it causes a failure of a flip-flop.

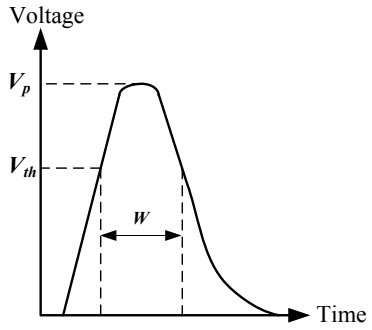

Figure 3.13: The SET pulse generated from a combinational circuit node.

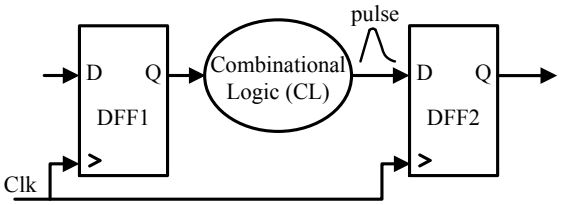

Figure 3.14: The synchronous pipeline architecture for failure rate model.

3.6.1.1 SET Failure Rate Model for Conventional Flip-flops

For an SET pulse to be sampled and cause a failure in a conventional flip-flop, it has to satisfy the following two conditions:

(a) The amplitude of the SET pulse should exceed the threshold voltage before the data is sampled at the input of the flip-flop;

(b) The amplitude of the SET pulse remains higher than the threshold voltage, while the input is being sampled.

The probability of a functional failure caused by a transient pulse is the product of the probability of the two conditions.

Probability of Condition (a): When an SET is generated, there can be two scenarios as shown in Figure 3.15. In Scenario (1), the amplitude of the pulse exceeds the threshold voltage of the input gate before the data is sampled at the rising edge of the clock. In Scenario (2), the amplitude of the pulse exceeds the threshold voltage after the data is sampled. Scenario (1) satisfies Condition (a), while Scenario (2) does not cause a failure as the SET cannot be sampled by the flip-flop.

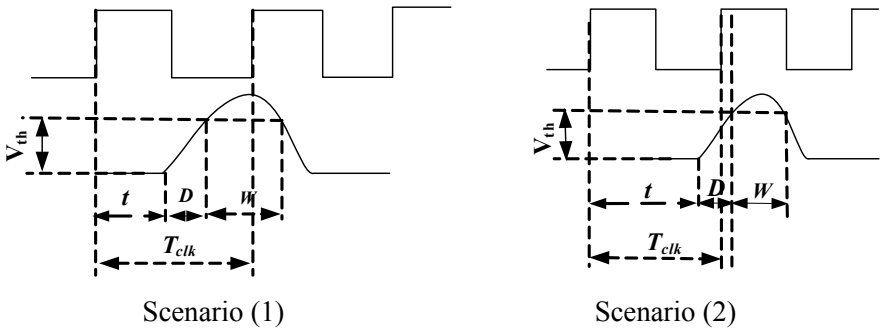

Figure 3.15: Timing Condition (a) of the transient pulse.

Let T_{clk} be the clock period. t denotes the time that the pulse appears at the input of the DFF2, which can be any time during the clock cycle, therefore t can be written as:

$$t = \alpha T_{clk} \tag{3.8}$$

where

$$0 \leq \alpha \leq 1$$

D denotes the time it takes for the pulse to reach the threshold voltage. It depends on the delay of the path the pulse has taken (logic + wire delay). The probability of Scenario (1), $g1$, can be written as:

$$g1 = Pr(T_{clk} \geq D + \alpha T_{clk}) = Pr(D + (\alpha - 1)T_{clk} \leq 0) \tag{3.9}$$

where Pr stands for the probability. The timing variables D and T_{clk} are functions of the physical layout and the supply voltage of the circuit. The latter are subject to random process variations [71] [72]. Based on the results and arguments in [71] [73, 74, 75, 76], it is reasonable to assume that these timing variables can be modelled by normal distributions. μ_D and $\mu_{T_{clk}}$ denote the mean values of D and T_{clk} respectively, while σ_D and $\sigma_{T_{clk}}$ denote the standard deviations. The closed form solution of Equation 3.9 can be obtained as a function of α [77], as follows:

$$g1 = \Phi\left(\frac{(\alpha - 1)\mu_{T_{clk}} + \mu_D}{\sqrt{\sigma_D^2 + ((\alpha - 1)\sigma_{T_{clk}})^2}}\right) \tag{3.10}$$

the Φ-function can be expressed in terms of the error function $erf(x)$, as follows:

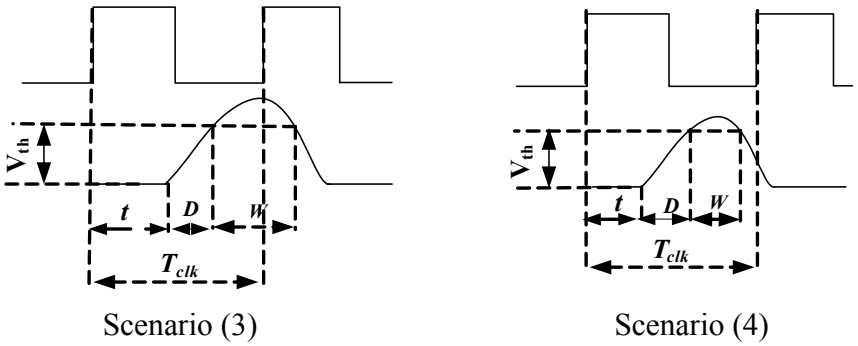

Figure 3.16: Timing Condition (b) of the transient pulse.

$$\Phi(x) = \frac{1}{2} + \frac{1}{2} erf(\frac{x}{\sqrt{2}}). \tag{3.11}$$

where:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-x^2}\, \mathrm{d}x. \tag{3.12}$$

Probability of Condition (b): If the transient pulse satisfies Condition (a), there are another two cases shown in Figure 3.16. Scenario (3) satisfies Condition (b), in which the amplitude of the pulse remains higher than the threshold voltage when the pulse is sampled. Scenario 4 does not cause failures since the amplitude of the pulse falls below the threshold voltage before the data is sampled. w denotes the width of the pulse, the time period for which the amplitude of the pulse exceeds the threshold voltage. The probability of Scenario (3) can be given as:

$$\begin{aligned} g2 &= Pr(T_{clk} \leq D + \alpha T_{clk} + w) \\ &= Pr(T_{clk} - \alpha T_{clk} - D - w \leq 0) \end{aligned} \tag{3.13}$$

According to the SET distribution results presented in [78, 79, 80], SET pulses can also be modelled by a normal distribution. Hence, similar to Condition (a), the closed form solution for Equation 3.13, as a function of α, is:

$$g2 = \Phi(\frac{(1-\alpha)\mu_{T_{clk}} - \mu_D - \mu_w}{\sqrt{\sigma_D^2 + \sigma_w^2 + ((1-\alpha)\sigma_{T_{clk}})^2}}) \tag{3.14}$$

where that μ_w and σ_w are the mean and the standard deviation of the width of the SET pulse, respectively.

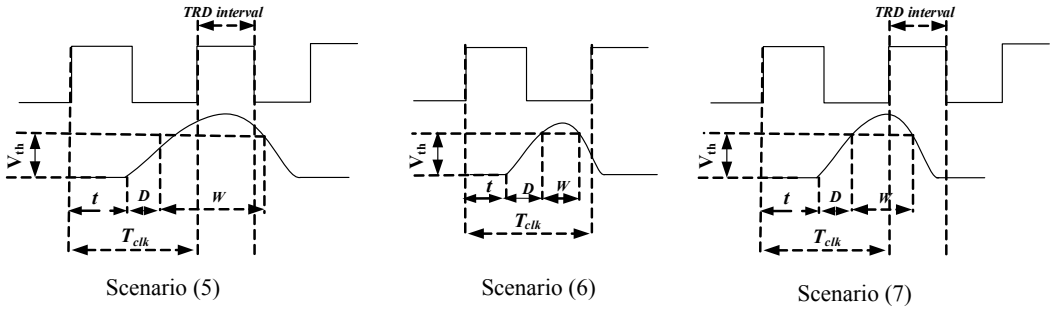

Figure 3.17: Timing Condition (b) of the SET for SETTOFF.

The failure rate of a conventional flip-flop in the presence of such a transient pulse is hence the product of $g1$ and $g2$, as follows:

$$\begin{aligned} Fr &= g1 \cdot g2 \\ &= \Phi\left(\frac{(\alpha-1)\mu_{T_{clk}} + \mu_D}{\sqrt{\sigma_D^2 + ((\alpha-1)\sigma_{T_{clk}})^2}}\right) \cdot \Phi\left(\frac{(1-\alpha)\mu_{T_{clk}} - \mu_D - \mu_w}{\sqrt{\sigma_D^2 + \sigma_w^2 + ((1-\alpha)\sigma_{T_{clk}})^2}}\right) \end{aligned} \tag{3.15}$$

3.6.1.2 SET Failure Rate Model for SETTOFF

Both versions of SETTOFF used the adapted TRD architecture to mitigate the SETs manifest at their inputs. The SET-tolerant capability of SETTOFF1 and SETTOFF2 are the same, therefore they produce the same failure rate in the presence of the same SET pulses. We consider the circumstance when only an SET is captured, yet not detected by the TRD architecture as a failure. The SETs that are captured but detected by the TRD architecture will be recovered by the architectural recovery mechanism. In other words, the failures caused by the detected SETs will be recovered and will not contribute to the SER of the system. Therefore they are not included in the failure rate of SETTOFFs.

As with a conventional flip-flop, an SET also has to satisfy two conditions to produce a failure in SETTOFFs. The first condition is the same as that for a conventional flip-flop, where the amplitude of the pulse has to exceed the threshold voltage before the data is sampled at the input of the SETTOFFs. However, the second condition changes since the adapted TRD architecture is incorporated, for detecting the sampled SETs. Figure 3.17 shows three scenarios that exist for the second condition for SETTOFF. Only Scenario (5) can cause a failure and satisfy the second condition. This is because in this case, the pulse is sampled and the width of the pulse also exceeds the TRD interval, which will let the pulse escape from the detection. In Scenario (6), the pulse is not sampled. In Scenario (7), the pulse is sampled, but the amplitude of the pulse falls below the threshold voltage before the end of the TRD interval. This will allow the

pulse to be detected by the TRD architecture. According to Scenario (5), the second condition requires the amplitude of the SET pulse to remain higher than the threshold voltage not only when it is sampled, but also until the end of the TRD interval.

Let the duty cycle of the clock be $\tau = T_{TRD}/T_{clk}$, where T_{TRD} denotes the TRD interval which is equal to the high phase of the clock. The probability of the second condition for both versions of the SETTOFF can be derived as:

$$\begin{aligned} g2' &= Pr(T_{clk} + \tau T_{clk} \leq D + \alpha T_{clk} + w) \\ &= Pr(\tau T_{clk} + T_{clk} - \alpha T_{clk} - D - w \leq 0) \end{aligned} \tag{3.16}$$

Similar to the derivation of the failure rate for the conventional flip-flop, the closed form of $g2'$ can be expressed by Equation 3.17, as follows,

$$g2' = \Phi\left(\frac{(1+\tau-\alpha)\mu_{T_{clk}} - \mu_D - \mu_w}{\sqrt{\sigma_D^2 + ((1+\tau-\alpha)\sigma_{T_{clk}})^2 + \sigma_w^2}}\right) \tag{3.17}$$

The SET failure rate of the SETTOFFs in the presence of the SETs can hence be derived by the product of $g1$ in Equation 3.10, and $g2'$ in Equation 3.17, as follows:

$$\begin{aligned} Fr' &= g1 \cdot g2' \\ &= \Phi\left(\frac{(\alpha-1)\mu_{T_{clk}} + \mu_D}{\sqrt{\sigma_D^2 + ((\alpha-1)\sigma_{T_{clk}})^2}}\right) \cdot \Phi\left(\frac{(1+\tau-\alpha)\mu_{T_{clk}} - \mu_D - \mu_w}{\sqrt{\sigma_D^2 + ((1+\tau-\alpha)\sigma_{T_{clk}})^2 + \sigma_w^2}}\right) \end{aligned} \tag{3.18}$$

3.6.2 SEU Failure Rate Evaluation Model

The SEUs that flip either the state stored in the master latch during the high clock phase, or the state stored in the slave latch during the low clock phase will cause a failure at 100% probability at the output of a D-type flip-flop. Therefore the failure rate of a conventional flip-flop due to SEUs is simply 100%.

As discussed in Section 3.3.1 and Section 3.4.1, for both versions of SETTOFF, the SEUs occurring in the master latch during the high clock phase will be addressed by the TRD architecture, hence they cannot cause failures. The SEUs that corrupt the slave latch during the low clock phase will be detected and corrected on the fly, by generating correction glitches. The correction glitch is not considered as a failure unless it corrupts the following stage. The failure rate caused by the SEUs occurring in the slave latch in the SETTOFFs therefore equals the failure rate caused by the correction glitches in the following stage. Similar to an SET pulse, the failure rate caused by a correction glitch

depends on the occurrence time of the glitch and its width, and can be derived using the SET failure rate model.

When the following stage is a conventional flip-flop, the failure rate caused by the correction glitch can be derived by Equation 3.15. When the following stage is a SETTOFF, the failure rate of the correction glitch can be obtain by Equation 3.18. Therefore, rather than producing a 100% failure rate in a conventional flip-flop, SEUs can only cause SETTOFF to produce a failure rate equalling to that is caused the induced correction glitches.

3.6.3 Discussion

It should be noted that the proposed SET and SEU failure rate models for both the conventional flip-flop and the SETTOFFs are not intended to calculate the accurate failure rates of the flip-flops. Nevertheless, the purpose is to provide a tool that can predict the impact of the particle strikes, and quantitatively compare the reliability of various sequential logic architectures.

On the other hand, the SET failure rate model for the SETTOFF can be used to quantitatively predict the SET-detection capability of the TRD architecture. Although it is claimed in Chapter 2, Section 2.3.4.2 that the TRD can detect all SETs when the TRD interval is big enough, such a generous TRD interval can be hard or impossible to achieve in modern microprocessors driven by high clock frequencies. This is because the clock period can be smaller than the pulse width, but the TRD interval cannot be greater than the clock period. Moreover, a bigger TRD interval introduces stricter timing constraints (see Chapter 2, Section 2.3.4.5, and Chapter 3, Section 3.5), hence may require more buffers to increase the shortest path delay of the combinational logic. The buffers will incur extra area and power overheads.

For non-safety-critical commercial applications, a zero SET failure rate may not be a prerequisite. A smaller TRD interval may be sufficient and desirable to satisfy the required trade-offs between the reliability and the error-tolerance overheads. The SET failure rate model can quantify different SET failure rates when different TRD intervals are applied, therefore can provide a valuable reference for choosing the optimum TRD interval at early design stages.

3.7 Experimental Setups and Comparative Evaluation Results

3.7.1 Experimental Methodology

Both SETTOFF1 and SETTOFF2 have been implemented [1] and tested in STMicroelectronics 65nm and 120nm technologies for verification and evaluation. The proposed error-tolerant architectures in both SETTOFFs were constructed as SPICE netlists in HSpice, using the transistor models from the two technology libraries. The SPICE netlist of a conventional D-type flip-flop was picked from each technology library to construct the main flip-flop in SETTOFF. The power consumption and performance (Clock-to-Q delay and setup time) of both SETTOFFs are then measured through SPICE simulations (The detailed setup of simulating the SPICE netlist for extracting power and performance numbers will be explained in Section 3.7.3 and Section 3.7.4, respecively). The cell area of the SETTOFFs are estimated and compared by counting the number of transistors that are required for constructing the cell. This is because the average transistor size for constructing both SETTOFFs is similar to the standard transistor size for building the standard cell in the library, so the number of transistors can reflect a fair comparison of the cell area.

The error-tolerant overheads of SETTOFFs are evaluated in both 65nm 120nm technologies. This enabled us to carry out fair comparisons with previous radiation-hardened techniques, since some of the previous techniques such as RazorI are implemented and tested in 130nm technology[2], while others are evaluated in 65nm technology. The evaluation results from 2 different technologies can also indicate how the cost-efficiency of SETTOFF architecture changes along with technology scaling trend.

On the other hand, the reliability of both versions of SETTOFFs are also evaluated using both fault-injection and simulation in SPICE, and the failure rate model proposed in Section 3.6. The reliability results are compared with the previous radiation-hardened techniques.

3.7.2 Area Overhead of SETTOFF

The area overhead of SETTOFF is estimated by the number of extra transistors required for constructing a SETTOFF architecture based on a conventional flip-flop. Table 3.2 summaries the area overhead of both SETTOFF1 and SETTOFF2. Since the error flip-flop in the TRD architecture can be easily shared by multiple bits, a single SETTOFF1

[1] The transistor-level design of the cells are constructed using the library device model, but the cells are not fabricated

[2] We consider the difference between 120nm and 130nm technology to be minor

Table 3.2: Area Overhead of SETTOFF1 and SETTOFF2

Radiation hardened Flip-flop	Extra Transistors Required	No. of transistors of the main flip-flop	Area overhead
SETTOFF1	48	32	150%
SETTOFF2	30	32	94%

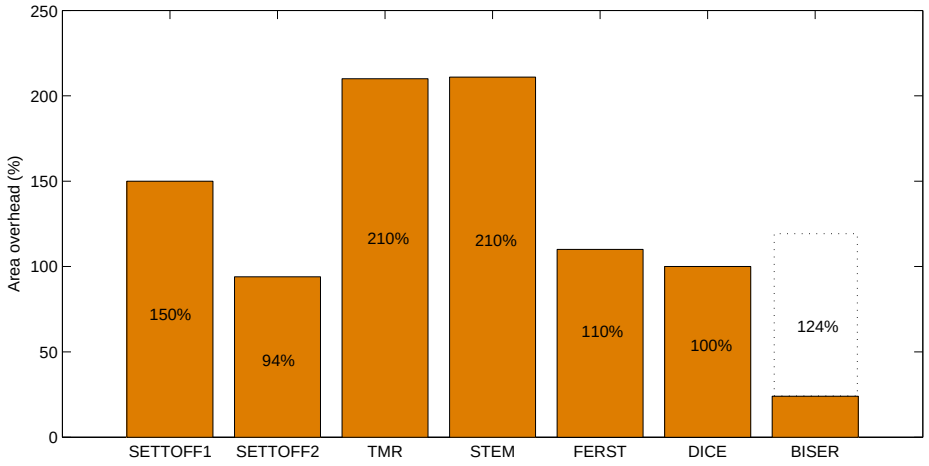

Figure 3.18: The area overhead comparisons, 65nm technology.

requires 48 extra transistors to construct, while a SETTOFF2 requires 30 extra transistors. Compared with a conventional flip-flop that consists of 32 transistors, SETTOFF1 and SETTOFF2 have 150% and 94% area overheads, respectively.

3.7.2.1 Area Overhead Comparisons

Figure 3.18 shows the comparisons of the area overheads for several radiation hardened cells. TMR and the SEM/STEM cell introduced in Chapter 2, Section 2.3.6, incur the biggest area overheads of over 200%. FERST and DICE rely on the duplication of the storage element (the cross-coupled inverter pair). As is reported in [33] and [22], both of them require area overheads of around 100%. It should be noted that BISER uses the Dual-Modular-Redundancy (DMR) architecture which uses a redundant flip-flop and some extra comparison circuitry to mitigate soft errors in sequential cells. BISER architecture induces an area overhead of 124%, compared to a conventional flip-flop architecture. However, the BISER architecture can be constructed by re-using the existing scan flip-flop to act as the redundant flip-flop in a system with scan chain. In this case, when not taking the area of the existing redundant flip-flop into account, the area overhead of BISER will only be caused by the additional comparison circuitry, which is

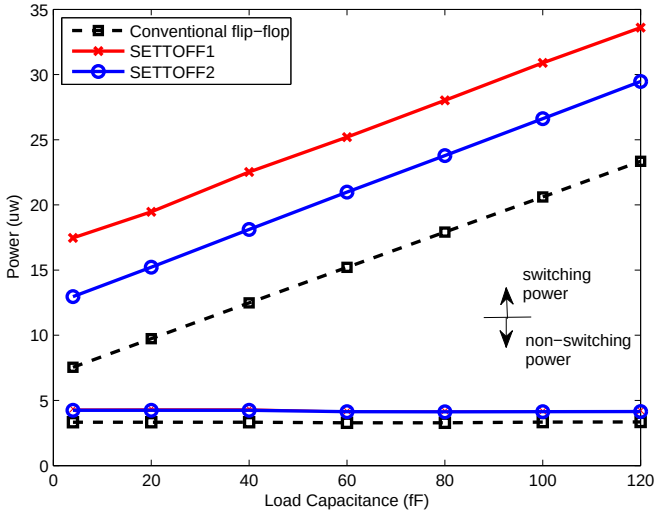

Figure 3.19: The power consumption of the flip-flops in 120nm technology.

24%. In addition to the cells shown in Figure 3.18, SETTOFF1 and SETTOFF2 have comparable area overheads with RazorI and RazorII flip-flops, which require 54 and 31 extra transistors, respectively.

3.7.3 Power Consumption Overhead of SETTOFF

The power consumption of SETTOFF1 and SETTOFF2 are measured in both 65nm and 120nm technologies, using SPICE simulations. The measurement results are compared with a conventional flip-flop with the same drive strength. The power consumption evaluation uses a 1.2V supply voltage, and a 185MHz clock to drive the flip-flops [3]. Both the clocks and the inputs of the flip-flops are driven by signals with 50ps transition times. A range of different load capacitances within the driving capability of the flip-flops are used for testing the power.

The power consumption of the conventional flip-flop and the two SETTOFFs in 120nm technology are shown in Figure 3.19. The upper traces show the average power consumed by a conventional flip-flop, SETTOFF1, and SETTOFF2 when the sampled data is switching. The lower traces show the power consumption of the three flip-flops when the sampled data is not switching. The power consumption overheads of both SETTOFF1 and SETTOFF2 compared with a conventional flip-flop in 120nm technology are shown in Figure 3.20. With a 10% activity rate, the extra power consumed by SETTOFF1 drops from 41.1% to 27.8% as the load capacitance increases from 4fF to 120fF, while the power overhead of SETTOFF2 ranges from 31.8% to 24.1% for the same conditions. The reason for this is because the redundant error-mitigation circuitry in both SETTOFFs

[3]The reason of choosing 185MHz clock is to make fair comparisons with the power consumptions of Razor flip-flops reported in [42] and [44].

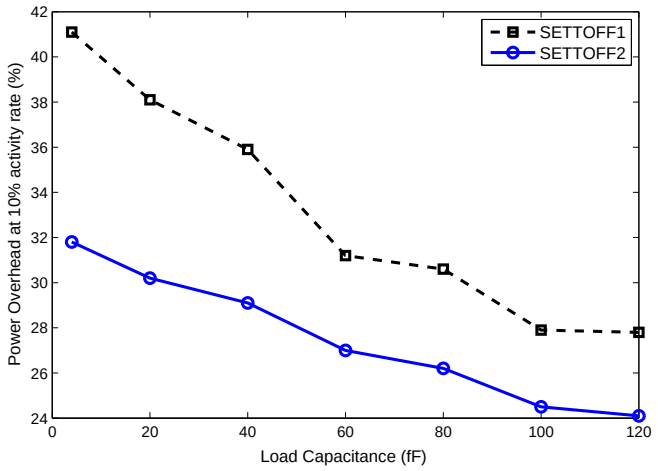

Figure 3.20: The power consumption overhead of the SETTOFFs in 120nm technology.

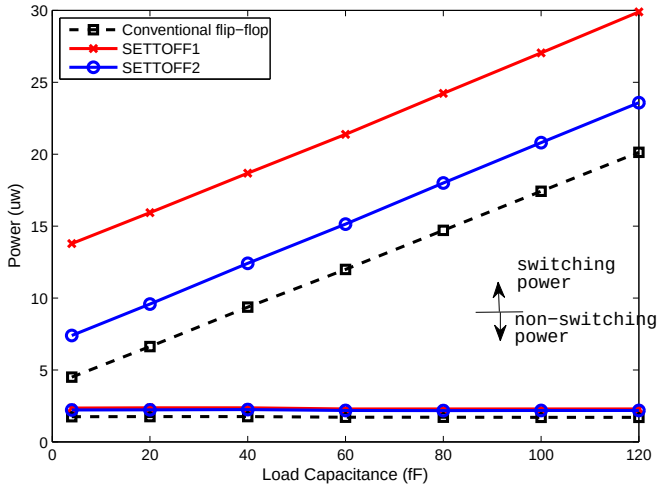

Figure 3.21: The power consumption of the flip-flops in 65nm technology.

do not work during normal operations. So it can be observed from Figure 3.19 that the extra power consumed by the error-mitigation circuitry in both SETTOFF1 and SETTOFF2 stays nearly constant as the load increases. Therefore, while the power consumption of the main flip-flop increases as load capacitances get bigger, the relative power overheads of SETTOFFs decrease.

A similar power consumption evaluation was also carried out in the 65nm technology. The same operating condition as in the 120nm technology is used. Figure 3.21 and Figure 3.22 show the power consumption and the power overhead of SETTOFF1 and SETTOFF2 over a library flip-flop in 65nm technology, respectively. As the load capacitance increases from 4fF to 120fF, the power overhead of SETTOFF1 ranges from 51.5% to 36.1%, while the power overhead of SETTOFF2 ranges from 30.8% to 26.5%

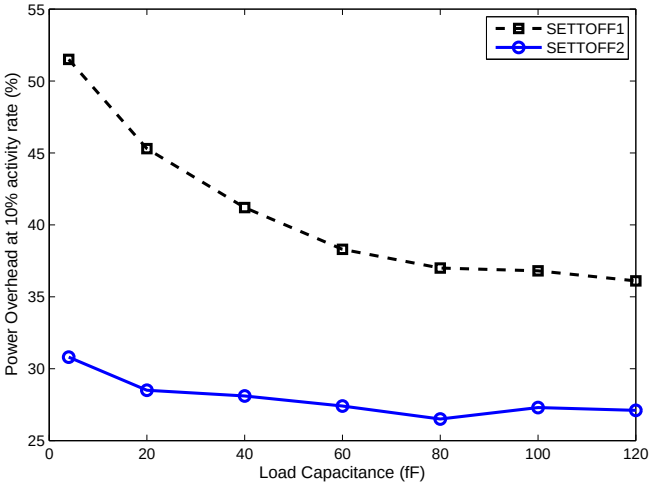

Figure 3.22: The power consumption overhead of the SETTOFFs in 65nm technology.

at a 10% activity rate. The average power consumption overheads for SETTOFF1 and SETTOFF2 in the 65nm technology are 40.8% and 28.0%, respectively.

3.7.3.1 Power Consumption Overhead Comparisons

Figure 3.23 shows the average power overhead comparisons of different hardening cells in 65nm technology. Similar to a TMR flip-flop, the SEM/STEM flip-flops incur over 200% power consumption overheads since they triplicate the main flip-flop and also add extra circuitry for error recovery. DICE and FERST rely on the duplication of the state-holding elements which consumes extra switching power even in normal operation. Therefore, their power overheads are around 100% [33], [30]. BISER incurs a power overhead of 126% compared to a scan flip-flop [35]. SETTOFF1 and SETTOFF2 incur relatively small power consumption overheads because the error-mitigation circuitry do not introduce extra switching power in normal operation.

According to [44], a RazorII flip-flop consumes 28.5% more power than a conventional flip-flop in 130nm technology [4] for the same operating conditions and activity rate that are used for evaluating the power consumption of SETTOFFs. RazorI flip-flop consumes 30.0% more power in the same conditions [5]. Thus, SETTOFF1 and SETTOFF2 requires comparable power overheads to the Razor flip-flops in 120nm technology.

[4] We consider the difference between 120nm and 130nm to be minor.
[5] The load capacitance used for measuring these results is not reported by the authors.

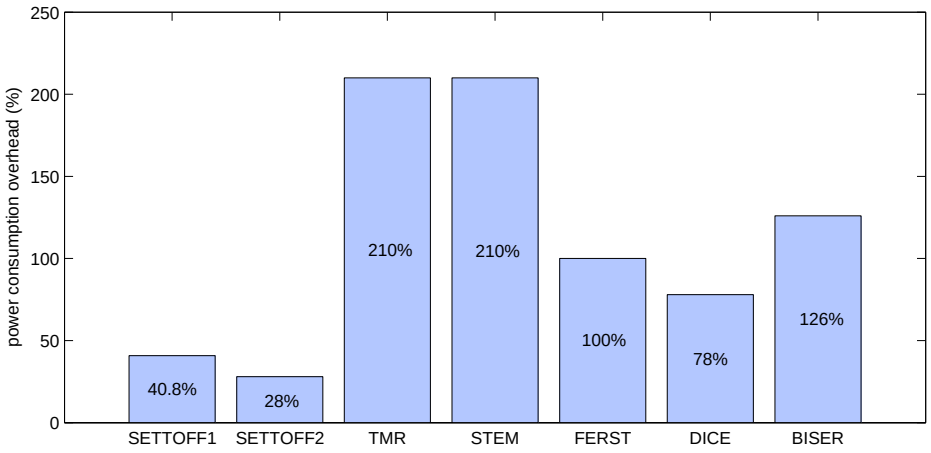

Figure 3.23: The power overhead comparisons, 65nm technology.

3.7.4 Delay Overhead of SETTOFF

The Clock-to-Q delay overhead of both SETTOFF1 and SETTOFF2 compared with a conventional flip-flop with the same drive strength is measured with different input transition times and load capacitances. The Clock-to-Q delays of SETTOFFs are measured from SPICE simulation based on the threshold and slew rate defined by the technology library, which are listed as follows [81]:

- Delays are defined as the time interval between the input stimulus crossing the threshold of 40% of Vdd for the rising signal and 60% of Vdd for the falling signal and the output stimulus crossing the threshold of 40% of Vdd for the rising signal and 60% of Vdd for the falling signal.
- The Transition times (slews) on input and output pins are defined as the time interval between the signal crossing 20% of Vdd and 80% of Vdd

The Clock-to-Q delay of both SETTOFFs derived from simulations are then compared with a conventional flip-flop with the same drive strength. The comparative results in 120nm and 65nm technologies are shown in Table 3.3 and Table 3.4, respectively. Compared to the conventional flip-flop, the delay overhead of SETTOFF1 is incurred by the correction XOR-gate in the signal path and the extra load it adds. SETTOFF2 moves the correction XOR-gate into the main flip-flop to replace the inverter driving the output Q. As discussed in Section 3.4.2, the delay of the correction XOR-gate equals that of the replaced inverter in normal operation. The increase of the Clock-to-Q delay in SETTOFF2 is only caused by the extra load introduced by the TD at the internal node N, and the TRD architecture at the output of the flip-flop.

In addition, the setup times of both SETTOFFs are measured in 65nm technology, and are compared with the setup time of the conventional flip-flop that from which the

Table 3.3: Clock-to-Q delay overhead in 120nm technology

	SETTOFF1		SETTOFF2	
Input transition times	4fF load	120fF load	4fF load	120fF load
5ps	37.6%	38.4%	31.6%	21.8%
33.3ps	34.9%	37.6%	25.8%	22.0%
100ps	29.6%	34.8%	24.8%	19.6%

Table 3.4: Clock-to-Q delay overhead in 65nm technology

	SETTOFF1		SETTOFF2	
Input transition times	4fF load	120fF load	4fF load	120fF load
5ps	45.0%	22.6%	20.8%	12.8%
33.3ps	41.0%	22.4%	18.4%	13.3%
100ps	33.3%	19.7%	15.2%	11.2%

SETTOFFs are constructed. The measurement is achieved by sweeping the transitions at the input of the flip-flop across the rising edge of the clock, and identifying the point where the flip-flop fails to sample the valid input transitions. The measurement results show that both SETTOFFs do not worsen the setup time compared with the conventional main flip-flop. With 50*ps* signal transitions for both the clock and the flip-flop input, the setup times of both SETTOFF1, SETTOFF2, and the conventional flip-flop are 40*ps* for the rising input transitions, and 30*ps* for the falling transitions.

3.7.4.1 Delay overhead Comparisons

Figure 3.24 shows the average Clock-to-Q delay overhead comparisons of different hardened cells in 65nm technology. A TMR flip-flop requires a majority voter added in the signal path, which incurs a delay overhead of around 70% in 65nm technology [33]. FERST and BISER both use a C-element at the output of the flip-flop. The C-element incurs a delay overhead similar to that of the TMR (i.e. 70%) [33]. Apart from the radiation hardened cells presented in Figure 3.23, RazorI and the SEM/STEM flip-flop add extra loads at the output of the flip-flop, which will also incur certain Clock-to-Q delay overheads [6]. RazorII has a slightly better CLK-to-Q delay since it is a latch based design [44].

3.7.5 Soft Error-Tolerance Capability of SETTOFF

The error-tolerance capabilities of both versions of SETTOFF are evaluated through both transistor-level simulations, and failure rate analysis using the model described in

[6] The delay overhead of Razor flip-flop and SEM/STEM cells was not reported by the authors

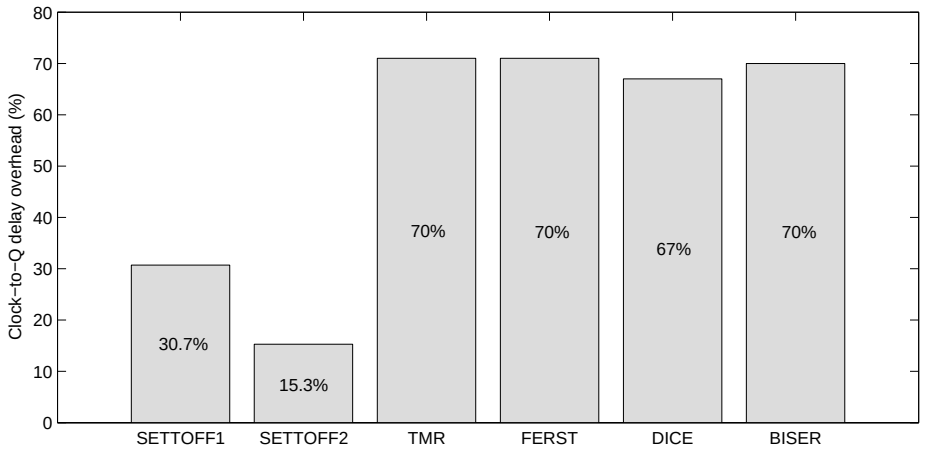

Figure 3.24: The Clock-to-Q delay overhead comparisons, 65nm technology [7].

Section 3.6. The evaluation results are then compared with the error-tolerance capabilities of other radiation hardened techniques.

3.7.5.1 Transistor-Level SET and SEU Injection and Simulation

The transient fault injection and simulations are carried out in 65nm technology for both SETTOFF1 and SETTOFF2. Independent current sources are used to simulate the collected charge induced by the particle strikes at circuit nodes. When sufficient charge is injected into a node, it will produce an SET or an SEU, depending on whether the node belongs to a combinational circuit or a storage element. A similar transient fault injection method has also been used in [33], [82]. Figure 3.25 shows the circuit schematic used for the fault injection and simulation. A symmetric clock with $2ns$ period is used for driving the main flip-flops.

For injecting SETs, an inverter is added at the input of the first SETTOFF. The current source $I1$ is placed at the output of the inverter to create SETs. The width of the SETs can be controlled by the injected current and the duration of the injection. A number of SETs with pulse widths of $800ps$ are injected at different time instances distributed across the entire TRD interval of the SETTOFF.

Another two current source, $I2$ and $I3$, are used for injecting SEUs into the master and slave latch of the main flip-flop, respectively. During the simulation, the time instances at which SEUs are injected into node $N1$ are swept across the entire high phase of the clock. Similarly, the time instances at which SEUs are injected into node $N2$ are also swept across the entire low phase of the clock.

[7] The delay overhead number for TMR, FERST, DICE, and BISER are obtained from the results published by their authors.

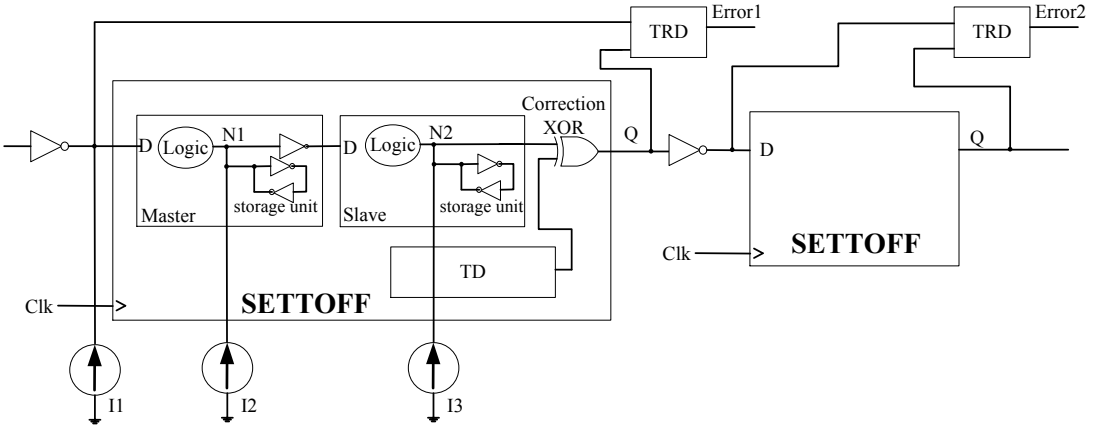

Figure 3.25: Soft error injection scheme.

Table 3.5: Transient fault injection simulation results for SETTOFF1 and SETTOFF2

	SETTOFF1	SETTOFF2
Number of injected SETs	50	50
Number of captured SETs	41	41
Number of detected SETs	41	41
Number of SEUs injected at node N1	50	50
Number of SEUs detected at node N1	50	50
Number of SEUs injected at node N2	50	50
Number of SEUs corrected at node N2	50	50
Number of correction glitches captured	3	1
Number of correction glitches detected	3	1

Table 3.5 shows the fault injection simulation results for both SETTOFF1 and SETTOFF2. All the captured SETs within the TRD interval are detected and error signals are recorded in the error flip-flops for both SETTOFF1 and SETTOFF2. Also, all the SEUs injected in nodes $N1$ are detected by the TRD architectures in both SETTOFFs. The SEUs injected in nodes $N2$ for both SETTOFFs are detected and corrected on the fly by their TD-based architectures. It should be noted that correction glitches are generated upon correction of the SEUs in node $N2$. However, only 3 correction glitches are captured by the following stage for SETTOFF1. All of them are detected by the TRD architecture in the second stage. For SETTOFF2, only 1 correction glitch is captured by the second stage. This is because the width of the correction glitches generated by SETTOFF2 is narrower than those are generated SETTOFF1. The captured correction glitch in SETTOFF2 is also detected by the following stage.

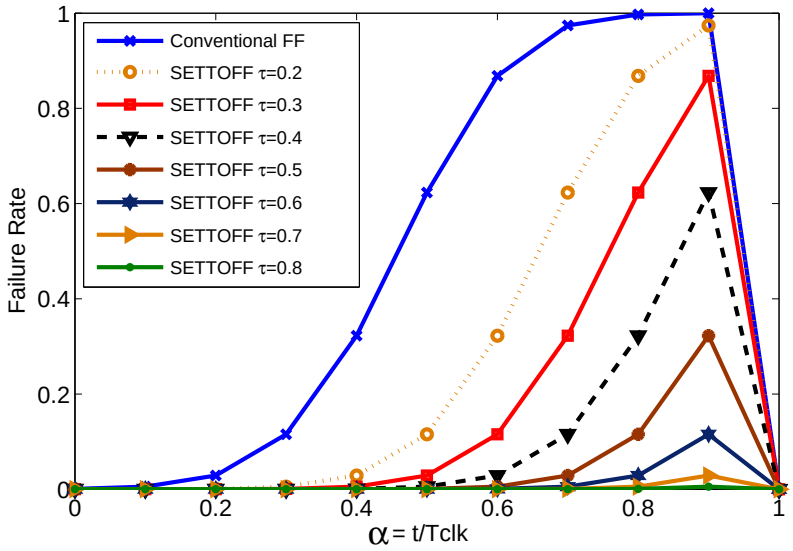

Figure 3.26: SET failure rate results.

3.7.5.2 SET Failure Rate Evaluation

The SET failure rates for the SETTOFFs are evaluated in the SET failure model proposed in Section 3.6.1, and are compared with the SET failure rate of a conventional flip-flop. Since both SETTOFFs used the same adapted TRD architecture to address SETs, their SET failure rates should be the same. The analysis uses 1GHz clocks to drive both the conventional flip-flop and the SETTOFFs.

According to the results presented in [83], the relative standard deviations of T_{clk} and D were both set to 10% to model parameter variations. The mean of the parameter $\mu_D = 10ps$, which is measured by using a combinational gate with a fixed drive strength and a flip-flop connected together. The SET width distribution results in 65nm technology, which are presented in [79], are used for the evaluation. The mean and the standard deviation of the pulse widths are derived from the distribution results, where $\mu_w = 530ps$ and $\sigma_w = 150ps$. Based on these numbers, the SET failure rates for the conventional flip-flop and SETTOFFs are derived from Equation 3.15 and Equation 3.18, respectively.

Figure 3.26 shows the comparative results of the SET failure rates between the conventional flip-flop and the SETTOFF driven by clocks with 1GHz frequency, but different duty cycles (τ). For both the conventional flip-flop and SETTOFFs, the SET failure rates are a function of α, which indicates the time instance when a transient pulse occurs during the clock cycle (see Equation 3.8). Obviously, SETTOFF reduces the failure rate over the conventional flip-flop, and as τ increases, SETTOFF becomes more reliable since the TRD interval becomes wider.

It can be calculated from Figure 3.26 that with different α values, the average SET failure rate of a conventional flip-flop driven by a 1GHz clock is 45%. SETTOFF reduces the

average SET failure rate to 4% with a 1GHz symmetric clock (i.e. $\tau = 0.5$, TRD interval = 500ps). It can further reduce the average SET failure rate to zero when the TRD interval reaches 800ps ($\tau = 0.8$).

3.7.5.3 SEU Failure Rate Evaluation

The analysis of the SEU failure rates for both the conventional flip-flop and the SETTOFFs are also carried out in 65nm technology. As is discussed in Section 3.6.2, the SEU failure rate for a conventional flip-flop is 100%. The SEU failure rates of both SETTOFFs are equal to the failure rates caused by the SEU-correction glitches generated from the TD-based architecture. In order to derive the SEU failure rate of both SETTOFFs, the widths of the correction glitches are measured under various PVT corners. The temperature variation ranges from 5°C to 45°C with a typical value of 25°C. The supply voltage of 1.2V, and transistor sizes vary from -10% to 10%. The measurement results show that the correction glitches of both SETTOFFs can also be modelled by a normal distribution.

For SETTOFF2, the mean of the width of the correction glitches $\mu_{w'} = 98ps$. The standard deviation $\sigma_{w'} = 33ps$. Based on these figures, the SEU failure rate of SETTOFF2 can be obtained by Equation 3.18. The results show that SETTOFF2 can reduce the SEU failure rate to 0 when using a 1GHz symmetric clock (500ps TRD interval). This is because the width of the correction glitch is much smaller than that of a typical SET pulse, so the TRD interval can easily cover such glitches even when they propagate and are captured.

For SETTOFF1, The mean and the standard deviations of the correction glitches are $\mu_{w'} = 153ps$ and $\sigma_{w'} = 50ps$, respectively. Again based on Equation 3.18, SETTOFF1 can also achieve a 0% SEU failure rate when using a 1GHz symmetric clock.

3.7.5.4 Reliability Comparisons

Table 3.6 compares the error-tolerance capability of SETTOFF with other previous techniques. The SET-tolerant capability of SETTOFF is similar to that of the FERST, STEM, and RazorII flip-flop. Other techniques cannot address SETs. Only SETTOFF, STEM, and the Razor flip-flops can address TEs, but both Razor flip-flops have limited soft error tolerant capabilities.

3.8 Concluding Remarks

Protecting the sequential elements (the flip-flops and latches) is the key for reducing the SER of general logic. This is not only because the SEUs occurring in the sequential cells

Table 3.6: Comparisons of the error-tolerance capability.

Capability	SETTOFF	TMR	DICE	FERST	BISER	STEM	RazorI	RazorII
SET	✓	×	×	✓	×	✓	×	✓
SEU	✓	✓	✓	✓	✓	✓	×	×
TE	✓	×	×	×	×	✓	✓	✓

contribute most to the SER, but also because only the SETs that are captured by the sequential gates can induce soft errors. Radiation hardened cells are a effective technique of protecting the sequential cells. However, most previous techniques either require large overheads, or can only provide limited error-tolerance capabilities. This makes them hard to use in commercial applications to achieve both a competitive performance and the reliability requirement at the same time.

This section proposed two versions of a novel radiation hardened architecture named SETTOFF, which can tolerate both SEUs and the captured SETs originating from the preceding combinational logic. Timing errors from the combinational gates can naturally be addressed as well. SETTOFF provides a better error-tolerance capability than most of the previous techniques. Meanwhile, SETTOFF minimises the error-tolerance overhead by separating the error recovery process into two levels (circuit-level and architectural-level). The evaluation results show that both versions of SETTOFF require less or comparable error-tolerance overheads than most previous hardened cells. On the other hand, since timing errors can also be addressed, SETTOFF can potentially be used to achieve an over-clocking system like STEM, or an aggressive DVS system like Razor.

In addition, a reliability metric, the failure rate evaluation model, is also proposed in this section. The model can quantitatively compare the reliability of various sequential architectures. The SET failure rate model can also quantify the reliability that a TRD architecture can provide by using different TRD intervals. It can be a valuable tool that can provide a reference for choosing the optimum TRD interval in early design stages, such that the best trade-off between the reliability and the implementation overheads can be achieved.

Chapter 4

The Self-Checking Register Architecture

As discussed in Section 1.6 of Chapter 1, apart from being expensive, the other drawback of most previous radiation hardening techniques is that they can only mitigate the errors occurring in the original circuitry, but cannot address errors occurring in the redundancies added for error-tolerance. In other words, they are not self-checking. The severity of this problem depends on the characteristic of the redundancies. The area and geometry of the redundancies determine the probability that the circuity are exposed to particles, while the critical charge determines the vulnerability of the circuit to particle strikes. On the other hand, if the unprotected redundancies are combinational logic, such as the encoding and decoding circuitry in an ECC or the majority voter in a TMR, they may produce SET pulses which can cause errors if captured in the following stage. If the redundancy is a retention element, such as the C-element in FERST or BISER, introduced in Chapter 2, Section 2.3.2 and Section 2.3.3, respectively, it can produce SEUs in a similar manner to a latch, and corrupt the whole cell directly.

The SETTOFF radiation hardened architectures proposed in Chapter 3 can mitigate both the SEUs corrupting the state stored in the flip-flop, and the SETs captured by the flip-flop. As demonstrated in Section 3.7 of Chapter 3, the SETTOFF architectures can provide a higher level of reliability with less or similar error-tolerance overheads compared to most of the previous techniques. However, similar to the previous techniques, a single SETTOFF is not self-checking as the added TRD- and TD-based architectures are not protected. This chapter describes a solution to this problem, by using a self-checker to mitigate the errors occurring in the redundancies in SETTOFF. Also, a self-checking register architecture is proposed to share the self-checking capability between multiple SETTOFFs at the register-level. The self-checking register is applied to protect the register file of microprocessors. The protected register file is implemented in 65nm technology, and is compared with the traditional ECC-protected RF.

This chapter is organised as follows: Firstly, in order to identify the problems caused by not being self-checking, the error vulnerability of the redundancies in a single SETTOFF is analysed in Section 4.1. Section 4.2 proposes a solution to the problem, which is to apply a self-checking architecture at the register-level. The experimental methodology and the evaluations of the self-checking architecture are described in Section 4.3. In Section 4.4, the proposed self-checking radiation hardened architecture is implemented to make a register file of a microprocessor robust. The robust register file is then compared with the ECC-protected register file implementation. Finally, the chapter is concluded in Section 4.5.

4.1 Soft Error Vulnerability Analysis for the Redundancy Circuities in SETTOFF

It has been discussed in Chapter 3 that the SETTOFF architecture can tolerate errors occurring in the original circuitry. However, SETTOFF is not fully self-checking. This section analyses the error vulnerability of the redundant circuitry in SETTOFF. As shown in Chapter 3, Figure 3.2 and Figure 3.7, the redundant circuitry in both SETTOFF1 and SETTOFF2 can be separated into 2 parts. The TRD architectures in Part I are the same in both SETTOFFs. Radiation particle strikes can induce SEUs in the error flip-flop, or SETs in the preceding comparator in the TRD architecture. Such SEUs or the SETs, if captured by the error flip-flop, can generate a false *Error* signal at the output of the TRD architecture. The false *Error* signal, however, only invokes an unnecessary replay execution when it appears during the TRD interval, but cannot corrupt the system. The unnecessary replay operations can incur an Instruction Per Cycle (IPC) overhead, thus affecting the performance of the processor. However it is not a prerequisite to protect the TRD architecture since it does not affect the correct system operation and therefore, the TRD architecture is not vulnerable to soft errors.

Part II of SETTOFF is the TD-based architecture. For both SETTOFF1 and SETTOFF2, shown in Figure 3.2 and Figure 3.7, respectively, the outputs of the TD-based architectures are the error signals generated by the transition detectors. As shown in Figure 3.6 and Figure 3.11, the transition detectors in both SETTOFFs are retention cells, which hold their states during the corresponding clock phases. However, the operating principles of the TD-based architecture in the two versions of SETTOFF are different. The error vulnerabilities of the TD-based architectures in SETTOFF1 and SETTOFF2 are therefore analysed separately.

The TD-based architecture in SETTOFF1 can be found in Figure 4.1. It consists of a detection clock generator, a transition detector, and a correction XOR-gate. The state stored in the internal node M of the TD can be switched by the transitions at the input of the TD. However, because node M is not driven by any inputs, it holds its state during

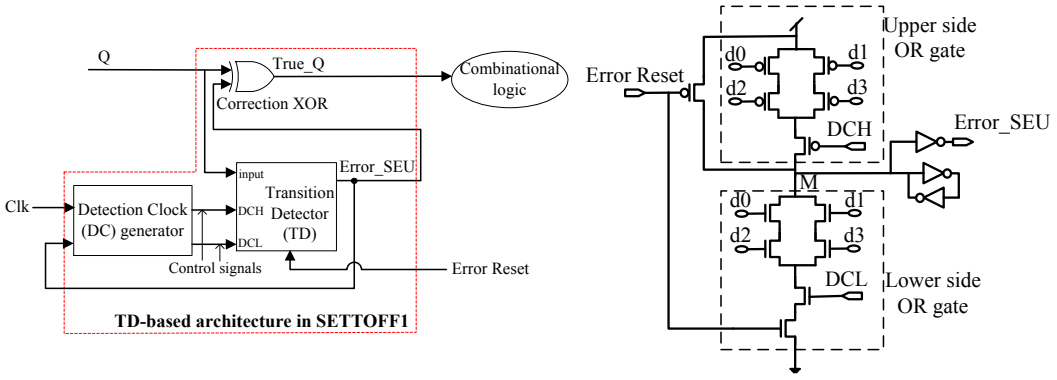

Figure 4.1: The TD-based architecture in SETTOFF1.

the entire clock period. High energy particle strikes at node M can flip the state, and induce SEUs at the output of the TD ($Error\_SEU$). On the other hand, SETs arising from other nodes of the TD, or the SETs occurring in the detection clock generator can also be captured by node M of TD, since the SET pulses may cause either of the dynamic OR-gates in TD to conduct momentarily. The captured SETs will flip the state stored in node M and hence flip the output of TD to generate a false $Error\_SEU$ signal. A false $Error\_SEU$ signal will corrupt the output of SETTOFF1 ($True\_Q$) directly through the correction XOR-gate. When any SETs or SEUs occurring in the TD-based architecture induce false $Error\_SEU$ signals during the TRD interval, the erroneous $True\_Q$ will be detected by the TRD architecture. As described in Chapter 3, Section 3.3, a replay recovery operation will be triggered upon the detection, which will reset the TD to recover the erroneous $True\_Q$. However, when the $Error\_SEU$ pin is corrupted during the TD interval, the erroneous $True\_Q$ will remain undetected, and therefore can propagate to the following stages. Consequently the TD-based architecture in SETTOFF1 is only vulnerable during the TD interval. The extra correction XOR-gate in the TD-based architecture can generate SETs at the output of the SETTOFF1. But similar to the SETs occurring in the original circuitry, these SETs will be detected by the SETTOFFs in the following stage if captured.

The TD-based architecture in SETTOFF2 shown in Figure 4.2 consists of an optimised transition detector and a correction XOR-gate. The optimised TD operates differently from the TD in SETTOFF1. During the TRD interval, the P-type transistor connecting node M is made conducting by the high clock signal. Therefore, node M is driven by the high voltage and the state held in M is forced at 1. Radiation particle strikes at any internal nodes of the optimised TD during the TRD interval cannot flip the state of node M, and hence the output of the TD ($Error\_SEU\_bar$) will stay free from any SEUs during the TRD interval. Nevertheless, SET pulses can arise from the internal nodes, which may propagate to the output of the TD during the TRD interval. Such SETs may further propagate through the correction XOR-gate and induce SETs at the output of

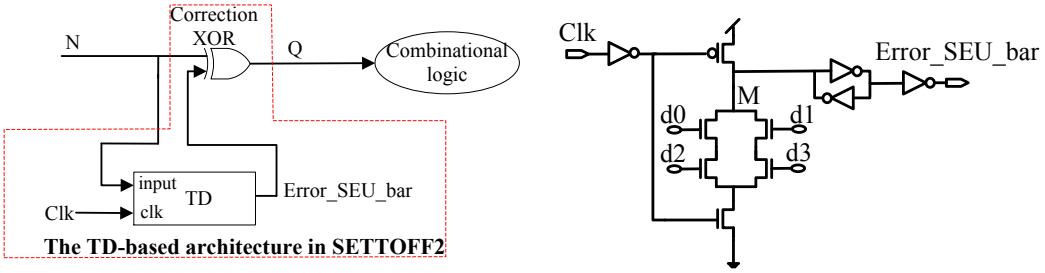

Figure 4.2: The TD-based architecture in SETTOFF2.

SETTOFF2. However, those SETs occurring during the TRD interval will be addressed by the TRD architecture. During the TD interval, the P-type transistor is blocked, thus the state-holding node M holds its state. In this case, node M can be corrupted by SEUs directly, or by the SETs occurring at other nodes that conduct the dynamic OR-gate in TD. The corrupted node M will produce a false $Error\_SEU\_bar$ signal at the output of TD, which will then corrupt the output of the SETTOFF2 through the correction XOR-gate. Therefore, the TD-based architecture in SETTOFF2 is also vulnerable to transient faults during the TD interval. Similarly to SETTOFF1, SETs arising from the correction XOR-gate will also be addressed by the TRD architecture in the following stages.

4.2 Self-Checking Radiation Hardened Register Architectures

As described in Section 4.1, both SETTOFF1 and SETTOFF2 are only vulnerable to errors occurring in the TD-based architecture during the TD interval. In order to realise the self-checking capability, a self-checker is required to address those errors that corrupt the output of the TD-based architecture during the TD interval. The self-checking mechanisms for both SETTOFF1 and SETTOFF2 can be both achieved by the same self-checker-based self-checking scheme described in Section 4.2.1. In order to minimise the overheads incurred by the self-checker, Section 4.2.2 proposes a sharing mechanism, which achieves the self-checking capability at the register-level by sharing the self-checker with multiple bits.

4.2.1 Key Concept of the Self-Checker

Since the output of the SETTOFF should be stable during the TD interval, soft errors that corrupt the output of the TD will propagate through the correction XOR-gate and induce a transition at the output of the SETTOFF. The idea of the self-checker is to

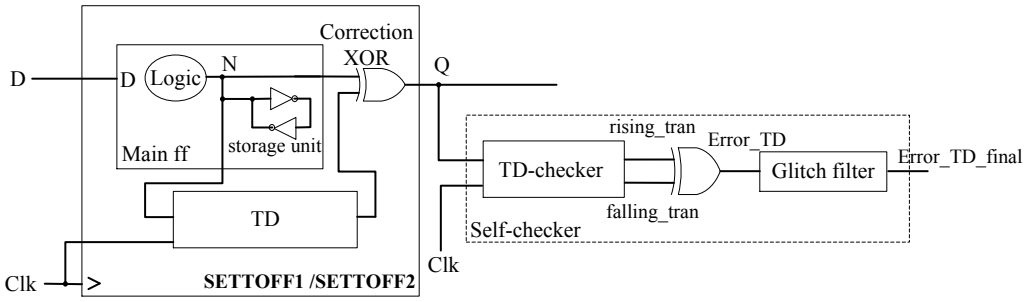

Figure 4.3: The self-checking SETTOFF architecture.

capture such illegal transitions at the output of SETTOFF, and to signal the errors in the TD-based architecture. However, it should be noticed that during the TD interval, correction glitches may also manifest themselves at the output of the SETTOFF with the corrections of the SEUs corrupting the main flip-flop (See Chapter 3, Section 3.3.1 and Section 3.4.1.). The self-checker therefore needs to distinguish between transitions caused by the correction glitches and illegal transitions caused by errors in the TD.

Figure 4.3 shows the architecture of the self-checking SETTOFF, which adds a self-checker to monitor the output of the SETTOFF during the TD interval. The self-checker consists of a TD-checker, an XOR-gate, and a glitch filter, to detect any transitions that are caused by the corrupted TD-based architecture in SETTOFF. The transistor level design of the TD-checker shown in Figure 4.4 is adapted from the transition detector built in SETTOFF2. The two delay chains constructed by the inverters and the transmission gates remain unchanged. The dynamic OR-gate for capturing the implicit pulses generated by the delay chains is separated into two branches, both driven by the system clock. During the TRD interval when the clock is high, node $M1$ and node $M2$ are charged by the high voltage, the TD-checker is disabled, and both of its outputs, $rising\_tran$ and $falling\_tran$, stay low. During the TD interval when the clock is low, the TD-based architecture in SETTOFF becomes vulnerable to soft errors. Therefore, the 2 branches are both enabled, to capture the pulses generated by the delay chain for the rising transitions and by the delay chain for the falling transitions, respectively. Any rising transitions occurring at the input of the TD-checker will discharge node $M1$ through transistors $d1$ and $d3$, and thus will be signalled at the output $rising\_tran$. Similarly, any falling transitions will assert the output $falling\_tran$ through the branch for the falling transitions.

The design of the TD-checker allows it to distinguish the correction glitches from the transitions caused by errors in the TD. This is because either a rising transition or a falling transition can only assert one of the two outputs of the TD-checker. However, a glitch consists of both a falling and a rising transition, thus will assert both outputs of the TD-checker. The two outputs, $rising\_tran$ and $falling\_tran$, are then XOR-ed

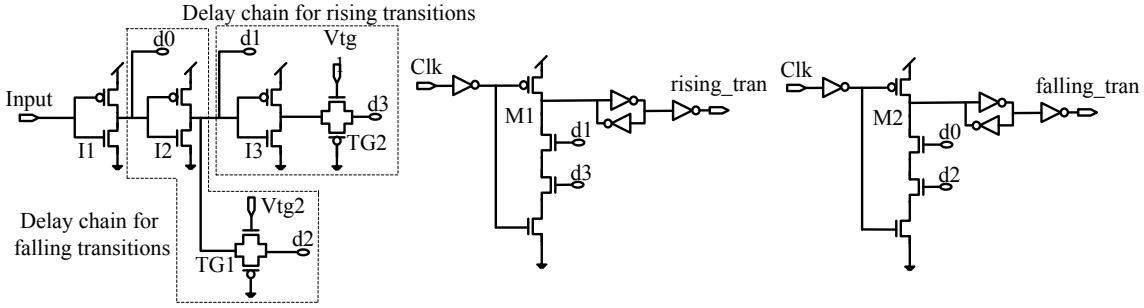

Figure 4.4: The transistor level design of the TD-checker.

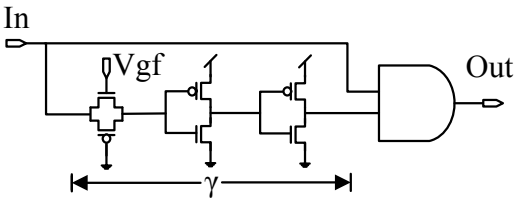

Figure 4.5: The transistor level design of the glitch filter.

together to generate the valid error signal ($Error\_TD$) for the errors corrupting the TD-based architecture. The $Error\_TD$ signal will only be asserted when only one of its input is asserted, which indicates an error. It will stay at 0 when both inputs are 0, or when both inputs are asserted due to the correction glitches.

Notice that there is a possibility that correction glitches occurring at node Q propagate through the TD-checker. This can be caused by the rising and falling transitions not asserting the $rising\_tran$ and $falling\_tran$ signals at exactly the same time instance. The time difference in asserting the $rising\_tran$ and $falling\_tran$ signals may lead to a positive glitch appearing at the $Error\_TD$ signal. A Glitch Filter (GF) is hence used to filter these positive glitches and generate the final error signal $Error\_TD\_final$. The transistor level design of GF is shown in Figure 4.5. The Transmission Gate (TG) and the two inverters connected in series create a delay chain. The input signal In is delayed by γ, which denotes the delay generated by the delay chain. In and the delayed version In_d are then AND-ed together, so that any positive glitches with widths no greater than γ are filtered out at the output. The value of γ can be set by tuning the Vgf of the TG. A normal transition occurring at the input of GF will simply propagate to the output.

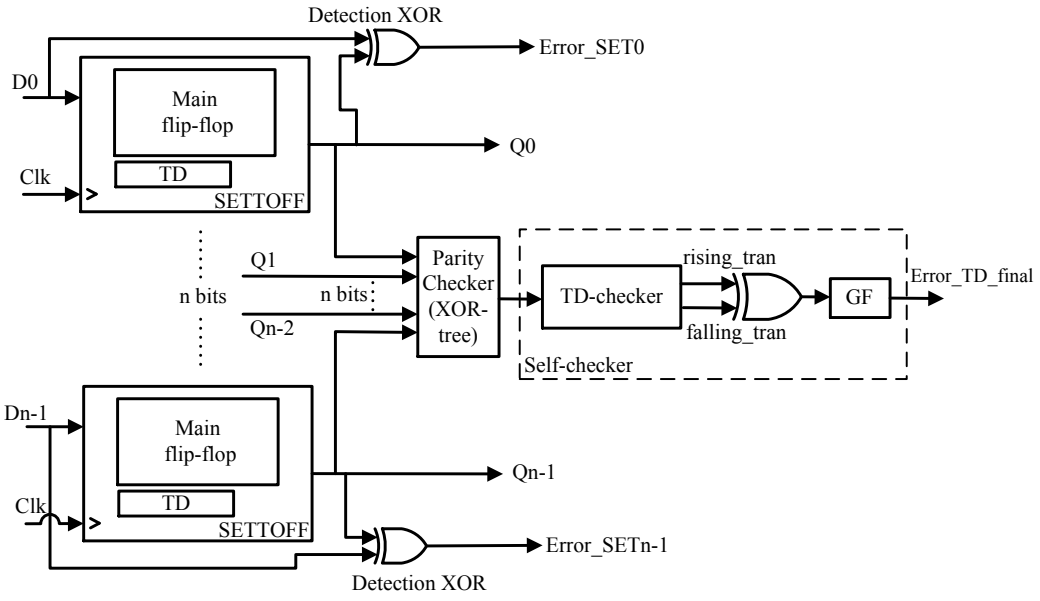

Figure 4.6: The self-checking radiation hardened register architecture.

4.2.2 Self-Checking Register Architecture

The self-checker added to the SETTOFF architecture incurs overheads in terms of area and power. Such overheads can be relatively big with respect to a single SETTOFF. In order to minimise the overheads introduced by the self-checking capability, we propose a mechanism to share the self-checker among multiple SETTOFFs in a register architecture. Figure 4.6 shows the architecture of an n-bit register constructed with SETTOFFs. Only one self-checker, as introduced in Section 4.2.1, is used to monitor all the outputs of all the SETTOFFs in the register through a parity checker. The parity checker is constructed by an n input XOR-tree, which connects the output of each SETTOFFs. Any illegal transitions occurring at the output of any single SETTOFF will change the parity of the total number of ones and zeros at the input of the parity checker, and therefore will be detected. Upon detection, the parity checker generates a transition at its output. Such transitions are then captured by the TD-checker and the XOR-gate, such that the $Error\_TD\_final$ signal will be asserted. It should be noted that the correction glitches occurring at the output of each SETTOFFs may also in turn induce glitches at the output of the parity checker. Therefore, the GF is again used to filter any positive glitches that may be induced by glitches at the output of the parity checker.

4.2.3 Circuit Implementation Issues

The sharing of the self-checker between multiple SETTOFFs causes certain implementation issues, which will be discussed. The parity checker constructed by the XOR-tree introduces a propagation delay from the output of the SETTOFFs to the input of the TD-checker. This delays the time that the transitions caused by the errors in TD take to reach the input of the TD-checker. Since the TD-checker is only enabled in the TD interval during which the TD-based architecture is vulnerable, the transitions that arrive at the input of the TD-checker after the TD interval will escape detection. A parity checker with a bigger delay indicates a higher probability that the transitions can escape. This problem can be addressed by reducing the number of SETTOFFs sharing the self-checker, so that a smaller parity checker can be applied. However, sharing the self-checker with less SETTOFFs requires more TD-checker, and thus may incur bigger area and power overhead. Therefore, the level of sharing the TD-checker represents the trade-off between the tolerant-capability for the errors occurring in the TD-based architecture, and the incurred error-tolerance overheads.

The delay of the parity checker also introduces timing constraints to the TRD interval. This is because valid transitions can occur at the output of the register on the rising edge of the clock during a write cycle of the register. The valid transitions will also be detected by the parity checker if they change the parity at the input of the parity checker. Such transitions would not normally be captured by the TD-checker as it is disabled during the TRD interval. However, if the delay of the parity checker exceeds the duration of the TRD interval, the valid transitions will arrive at the TD-checker during the TD interval, and will be falsely signalled as errors. To prevent such situations from happening, the delay of the parity checker must be smaller than the TRD interval.

In a register architecture constructed by SETTOFFs, the error flip-flop in the TRD architecture can be shared by all the bits. As shown in Figure 4.6, all the $Error\_SET$ signals from each bits of the register can be OR-ed together and fed into a shared error flip-flop. Any SETs captured by any bits inside the register are all recorded in the error flip-flop, which can be used to trigger an architectural replay recovery operation. However, the OR-tree used for sharing the error flip-flop introduces an extra delay (besides the comparators constructed by the XOR-gates) for the error signals to reach the error flip-flop. When implementing the TRD architecture at the register-level, the delay of the OR-tree can be regarded to as part of the delay of the comparator, as follows:

$$D_{comp} = D_{XOR} + D_{OR-tree} \tag{4.1}$$

where D_{XOR} is the delay of the comparator constructed by the XOR-gate, and $D_{OR-tree}$ denotes the delay of the OR-tree used for sharing the error flip-flop. Equation 4.1 can

then be combined with the other TRD constraints explained in Chapter 3, Section 3.5, to construct the TRD architectures at the register-level.

Notice that the extra delay introduced by the OR-tree affects the delay element δ that is required for covering SET pulses with certain widths. However, it does not affect the shortest path delay of the preceding combinational logic required for satisfying the TRD constraints. In other words, the delay of the OR-tree, and hence the sharing of the error flip-flop does not increase the number of buffers that need to be inserted to satisfy the shortest path constraint if required.

4.2.4 Recovery Mechanisms for the Errors Detected by the Self-Checker

The errors detected by the self-checker are the ones that corrupt the output of the TD-based architecture in SETTOFF. The basic idea for recovering these errors is to reset the TD-based architecture before the errors contaminate the forwarding logic. In this section, two potential recovery mechanisms for the SETTOFF2-based self-checking architecture are introduced. Both of them can be realised by combining with architectural-level operations when using the proposed self-checking architectures to protect the pipeline of a microprocessor. The recovery mechanism for the SETTOFF1-based self-checking architecture can also be achieved based on the same concept, but is not explicitly described.

4.2.4.1 Architectural Replay-Based Recovery Mechanism

This first recovery mechanism can be achieved by using the replay recovery operation similar to that for the error detected by the TRD architecture (See Chapter 3, Section 3.2). The $Error\_TD\_final$ signal can be used to trigger the pipeline replay operation in the following cycle, before the erroneous register output contaminates the forwarding logic. The replay operation will re-execute the latest instruction that writes to the corrupted register. During the re-write cycle, all the erroneous TDs in each bit of the register will be reset, hence the bit-flip errors at the output of the register will be recovered. It should be noted that with this recovery mechanism, the positive glitches in the $Error\_TD$ signal caused by the correction glitches at the register output can be tolerated. This is because such glitches can only trigger an unnecessary replay operation if they are captured by the following clock edge, but cannot corrupt the correct system operation. The glitch filter, therefore, can be removed from the self-checker when the replay recovery mechanism is applied. The details of the system-level implementation of this recovery mechanism can be found in Chapter 5, Section 5.3.2.

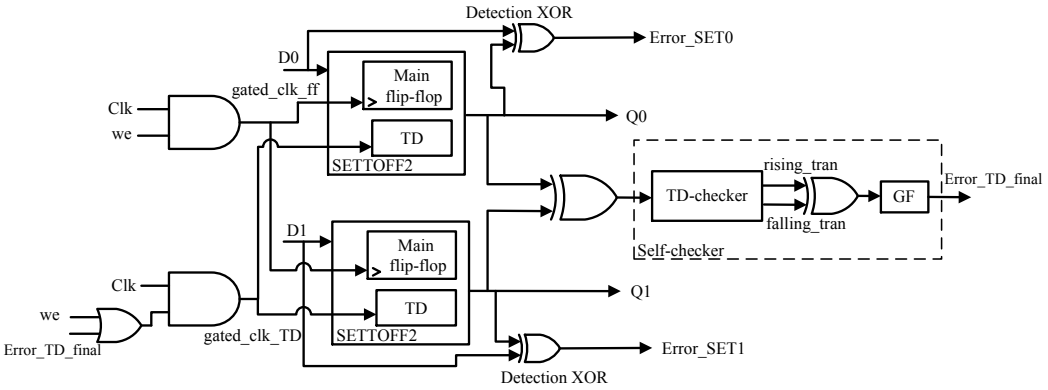

Figure 4.7: The clock-gating based recovery mechanism for the self-checking architecture.

4.2.4.2 Clock-Gating-Based Recovery Mechanism

As discussed in Chapter 3, Section 3.2, the pipeline replay function normally already exists in modern microprocessors to support speculative operations. Therefore, the replay-based recovery mechanism is relatively easy and cheap to achieve. However, the pipeline replay involves pipeline flush and pipeline re-executions, which cost extra cycles. The second recovery mechanism offers a quicker recovery for the errors detected by the self-checker, by using a clock-gating based architecture. An example of using the second recovery mechanism to protect a 2-bit register is shown in Figure 4.7. This mechanism can be applied to the SETTOFF2-based self-checking architecture combined with a clock-gating based flip-flop hold architecture. When an error that corrupts one of the TD-based architectures is detected by the self-checker, the asserted $Error\_TD\_final$ signal will trigger a pipeline stall to stall all the pipeline registers for one clock cycle. The disabled write enable signal, *we*, will gate the clock and force the register to hold during the stall cycle. Meanwhile, unlike the single SETTOFF2 for which the TD will also be gated during the hold cycle (See Chapter 3, Section 3.4.1), this recovery mechanism feeds the asserted $Error\_TD\_final$ signal back into self-checking register to generate a separate clock ($gated\_clk\_TD$) to drive the TD in each SETTOFF2. Therefore, the asserted $Error\_TD\_final$ signal will pass the system clock to the TDs, such that all the TDs will be reset by the high clock phase during the stall cycle. The detected errors in the TD and the erroneous output of the register will then be recovered. Normal execution resumes after the stall cycle so that the error-free data will propagate. Notice that the clock-gating-based recovery mechanism only requires one extra cycle, but it is more complicated and sacrifices more area and power consumption than the replay-based recovery mechanism.

4.2.5 Analysis for the Error-Tolerance Capability of the Self-Checking Register

The soft error vulnerability of the SETTOFF architecture has been discussed in Section 4.1. This section analyses the reliability of the SETTOFF-based self-checking register in various error situations. Table 4.1 summarises the types of errors that may occur in the proposed register, whether the errors can cause erroneous outputs, whether they can be tolerated, and the ways in which they are tolerated. The radiation induced transient faults affecting the register can be categorised into 5 types: 1, the captured SETs originated from preceding combinational logic. 2, the SEUs corrupting the output of the main flip-flop. 3, the errors corrupting the TRD architecture. 4, the errors corrupting the TD-based architecture. 5, the errors corrupting and the self-checker.

The captured SETs and the SEUs in the main flip-flop can induce erroneous outputs at the register, but they can be tolerated by the TRD and TD-based architectures, respectively. As discussed in Section 4.1, the errors corrupting the TRD architecture can induce a false *Error* signal, which only invokes an unnecessary replay operation, but does not damage the output of the register. Therefore, these errors do not need to be addressed. The errors corrupting the TD-based architecture during the TD interval affect the register output through the correction XOR-gate, thus they are mitigated by the self-checker. Notice that the the self-checker can also be affected by radiation strikes. SEUs can arise from the state-holding nodes, $M1$ and $M2$ of the TD-checker, which will invert either of its outputs, $rising\_tran$ or $falling\_tran$, and generate a false $Error\_TD\_final$ signal (See Figure 4.4). However, similar to the errors in the TRD architecture, the false $Error\_TD$ signal does not corrupt the register output since it can only invoke an unnecessary recovery process to reset the TDs in each bit. Therefore, the self-checker do not need to be protected either.

Table 4.1: Error-tolerance capability analysis of the self-checking register.

Error type	Damage to output	Tolerated	Means
captured SET	Yes	Yes	TRD arch.
SEU corrupting the main ff	Yes	Yes	TD-based arch.
Errors corrupting the TRD arch.	No	No	none
Errors corrupting the TD-based arch.	Yes	Yes	self-checker
Errors corrupting the self-checker	No	No	none

The built-in TRD and TD-based architectures in each SETTOFFs in the register allow the proposed register to tolerate an unlimited number of Multiple-Bit Upsets (MBUs) corrupting the outputs of multiple main flip-flops. The shared self-checker can also address MBUs corrupting multiple TD-based architectures during the same cycle. This is because the first corruption will be detected, and the recovery operation will then reset all the TDs in the register. However, if an even number of upsets simultaneously corrupt

multiple TDs, they can escape detection since the parity of the register output will not change in such situations. On the other hand, dangers can happen when MBUs corrupt both an architecture and the architecture that protects it during the same cycle. The SEUs in the main flip-flop, for instance, are protected by the TD-based architecture. However, if two upsets each corrupt the main flip-flop and the TD-based architecture during the same cycle, a recovery operation could be invoked to recover the upset in the TD-based architecture, while the upset in the main flip-flop would remain. The possibility of the occurrence of such MBUs can be reduced by applying certain layout approaches to space the vulnerable node pairs. The SBU- and MBU-tolerance capabilities of the self-checking design are demonstrated by circuit-level fault-injections and simulations, described in Section 4.3.3.

The MBU-tolerance capability of the self-checking register give it an advantage over an ECC-based register protection architecture, which requires noticeably larger overheads to correct MBUs (See Chapter 2, Section 2.5.2). The two architectures are both implemented in 65nm technology, and are compared. The details of this are given in Section 4.4.

4.3 Experimental Setups and Evaluations for the Self-Checking Radiation Hardened Register

4.3.1 Experimental Methodology

A 32-bit self-checking register based on SETTOFF2 has been implemented in 65nm technology for verification and evaluation. The SPICE netlist of the 32-bit self-checking register is constructed using the transistor models from the 65nm library, and is used for evaluating the power and performance of the register through SPICE simulations. The power and delay measurements are based on the threshold and slew rate defined by the technology library (see Chapter 3, Section 3.7.4 for detail). The supply voltage used for the evaluation is 1.2V. A 185MHz symmetric clock is used for driving both the register (with the positive edge), and the error flip-flop (with the negative edge) of the TRD architecture. The TRD interval is hence equal to 2.5ns after taking the setup time of the error flip-flop and the delay of the comparator into account. For a 32-bit register, 5 levels of gate delay are required for the XOR-tree that constructs the parity checker. The maximum propagation delay of the parity checker is 210ps, which is smaller than the TRD interval. The timing constraints of the self-checker discussed in Section 4.2.3 are therefore satisfied.

The maximum width of the positive glitches caused by the SEU-correction glitches presented at the signal $Error\_TD$ is 90ps under normal conditions. As reported in Chapter 3, Section 3.7.5.3, the standard deviation of the width of the correction glitches

is 33ps when taking PVT variations into account. The delay, γ, of the delay chain in the glitch filter (See Figure 4.5) is therefore set to 130ps, to ensure all the positive glitches appearing at signal $Error\_TD$ are filtered out at node $Error\_TD\_final$.

4.3.2 Implementation Overheads

The power consumption of the self-checking register was compared with a conventional register implemented with the same operating conditions and drive strength. With a 10% activity rate for a single bit, the average power overhead of the proposed register is 33%, which is only a 5% increase over the SETTOFF2 without the self-checking capability. (As reported in Chapter 3, Section 3.7.3, under the same conditions, the average power consumption overhead of SETTOFF2 in 65nm is 28%) In terms of area, a single SETTOFF2 requires 30 extra transistors. The proposed register only adds one self-checker, shared by each bit, thus the area overhead increase is insignificant. Comparing to a conventional register constructed from flip-flops with 32 transistors each, the area overhead of the 32-bit self-checking register is close to 136%. Since the self-checker is not added to the signal path of the register, the delay overhead of the register is comparable to that of a single SETTOFF2, with an average value of 16.5%.

4.3.3 Reliability Evaluations and Comparisons

The current source-based fault-injection mechanism introduced in Chapter 3, Section 3.7.5.1 is used for verifying the reliability of the self-checking register. Since the reliability of the single SETTOFF2 has been evaluated in Chapter 3, Section 3.7.5.1, this section focuses on simulating the transient faults occurring in the redundant circuitry of the proposed register, and the MBUs occurring inside the register.

The redundancies in the register are separated into 3 parts: the TRD architecture, the TD-based architecture, and the self-checker. As described in Section 4.2.5, the TRD architecture and the self-check are not vulnerable to radiation strikes since soft errors occurring in them cannot affect the output of the register. Therefore, transient fault injection and simulation is only carried out in the TD-based architectures in each SETTOFF. Because only the errors flipping the state held by the TD during the TD interval can corrupt the output ($Error\_SEU\_bar$) of the TD-based architecture in SETTOFF2, an independent current source is used to inject SEUs to node M of the TD (See Figure 4.2) during the TD interval. 30 SEUs are injected into the TDs in each SETTOFF2 in the register, thus a total of 940 errors are simulated. The injection times of the SEUs are evenly distributed during the TD interval. The simulation results show that out of the 940 SEUs, 884 are detected by the self-checker, and only 56 errors (6%) escaped. As discussed in Section 4.2.3, the escaped ones are caused by the delay of the parity checker, which causes the SEUs to reach the self-checker outside the TD interval. The

Table 4.2: Comparison of error-tolerance capability and overheads for 32-bit registers.

	Proposed Reg	TMR	ECC	FERST	BISER	STEM	DICE
Area overhead	136%	210%	206%	110%	24%	210%	100%
Power overhead	33%	210%	160%	100%	126%	210%	78%
Delay overhead	16.5%	70%	1 cycle	70%	70%	-	67%
SET-tolerance	YES	NO	NO	YES	NO	YES	NO
SEU-tolerance	YES	YES	YES	YES	YES	YES	YES
TE-tolerance	YES	NO	NO	NO	NO	YES	NO
Self-checking	YES	NO	NO	NO	NO	NO	NO

problem can be addressed by using more self-checkers to reduce the number of the SETTOFF2s sharing them, since this would allow a small parity checker with fewer levels of XOR-gates to be applied.

In order to validate the MBU-tolerant capability of the proposed register, the transient fault injection and simulation mechanism described in Chapter 3, Section 3.7.5.1 is applied to inject multiple faults to corrupt multiple bits of the register simultaneously. The simulation results demonstrate that all the MBUs are either individually corrected on the fly at the outputs of the corrupted SETTOFF2s, or detected by the TRD architectures incorporated in each bit.

Table 4.2 compares the implementation overheads and error-tolerance capabilities of several techniques for protecting a 32-bit register in 65nm technology. Notice that ECC is implemented with the SEC-DED (Single Error Correction Double Error Detection) coding which requires 7 redundant bits for a 32-bit register. The delay overhead of ECC is big due to the large decoding block, therefore an extra cycle may be required to reload the register for error correction. Due to the C-element added in the signal path, the delay overhead of FERST and BISER is similar to that of TMR. The area overhead of BISER is small since it uses the existing scan flip-flops as duplications. STEM uses a variant of TMR which removes the error correction from the signal path. The delay overhead of STEM is small [1]. Compared with other techniques, the proposed register incurs fewer comparable overheads. In terms of reliability, only the proposed register and STEM can tolerate both SETs, SEUs, and TEs. In addition, only the proposed register has a self-checking capability. Errors occurring in the redundancy circuit of other techniques may either corrupt the cells directly (e.g. FERST and BISER), or produce SET pulses (e.g. TMR, ECC, and STEM).

[1] The delay overhead of STEM was not reported by the authors in [41].

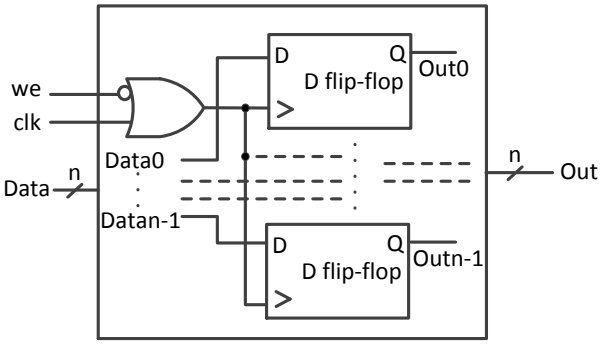

Figure 4.8: An n-bit register for constructing the RF.

4.4 Radiation Hardened Register File Implementations

As introduced in Chapter 2, Section 2.5.2, the Register File (RF) is one of the most vulnerable blocks in the microprocessor. The RF stores intermediate execution results and is frequently accessed. Efficiently protecting the RF is crucial for reliable microprocessor designs. Traditionally, the RF is protected by ECCs, which can introduce large performance and power overheads since the ECC-bits need to be calculated during each RF access. The SETTOFF2-based self-checking architecture proposed in this chapter could offer a new solution for soft errors in the RF. In order to validate the efficiencies of the proposed self-checking architecture applied to the RF, this section describes an implementation of an RF protected by the self-checking architectures in 65nm technology. The reliability and error-tolerance overheads of the protected RF are compared with the traditional ECC-based RF protection mechanism.

4.4.1 The Original Register File Architecture

The architecture of the original RF used for both robust implementations is introduced here. The RF is constructed from a number of registers with certain write and read logic. Figure 4.8 shows the architecture of an n-bit register for constructing the RF. The write enable signal (*we*) controls the write and hold operations of the register, by propagating or gating the clocks driving each D flip-flop in the register. The architecture of an RF consisting of m registers is shown in Figure 4.9. We implemented an RF with only one write port and one read port for evaluation. Notice that the actual RF in a microprocessor may require multiple write or read ports such that multiple registers can be written to or read from during one cycle.

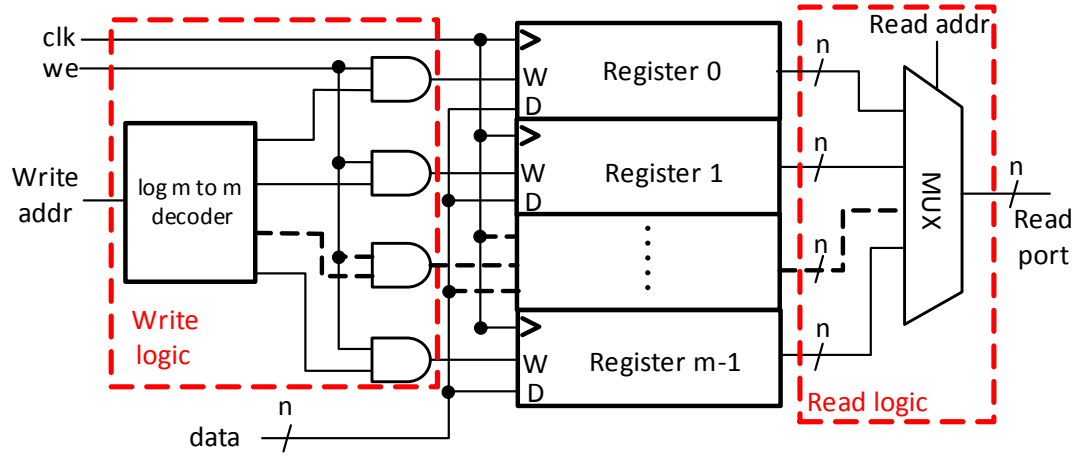

Figure 4.9: An original RF with 1 write port and 1 read port.

Write and read logic are required for each write and read port in the RF. For writing a register, a decoder is used to decode the write address and determine which register to write to. During a write operation, the *we* signal is asserted. A register number is fed into the decoder, which will then assert only the write line of the register to write to through the AND-gate. All the other write lines will be gated by the outputs of the decoder. The clock and the data feeding into each register are simply the ones to the RF. A big multiplexer is required for each read port of the RF, to determine the register to read from based on the read address. The multiplexer is an m-to-1 multiplexer, n-bits wide, where n is the number of bits in each register.

4.4.2 The Radiation Hardened RF Implementations

Based on the basic RF architecture shown in Figure 4.8, two robust 32-bit RFs, each containing 32 registers were implemented in 65nm technology. The first radiation hardened RF is protected by the proposed self-checking architecture. The protection is achieved by incorporating the self-checking register (Figure 4.6) into each register of the RF. In order to minimise the overhead incurred by the TRD architecture in the proposed register, only the TRD architecture for one register is implemented and is shared by the 32 registers in the RF. The architecture of the RF protected by the proposed self-checking register is shown in Figure 4.10. The write and read logic are identical to those of the original RF. Only 32 comparators (XOR-gates) are used for comparing the inputs and outputs of a 32-bit register. The error signals are OR-ed together and fed into a shared error flip-flop to record the error. A multiplexer is used for selecting the outputs of the register which is updated during the write cycle of the RF. The output data is then fed into the shared TRD architecture for SET detection. The shared error flip-flop is simply driven by the negative edge of the clock to produce the delay element δ, which is equal to the positive phase of the clock.

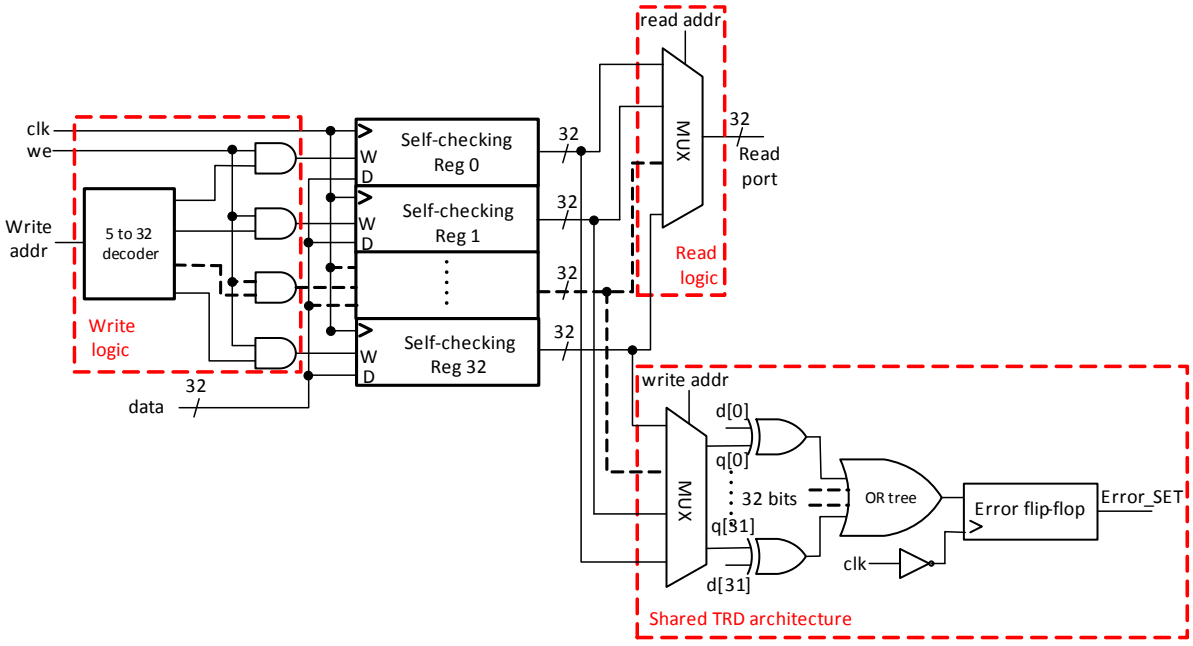

Figure 4.10: The self-checking register-protected RF.

It should be noted that the extra multiplexer introduces an extra delay for the outputs of the register to reach the TRD architecture. Such an extra delay is equivalent to an increase of the Clock-to-Q delay of the register. Therefore, the extra delay does not affect the TRD detection capability as long as it satisfies the TRD constraint specified by Equation 3.6 in Section 3.5, Chapter 3.

The second robust RF is protected by the ECC technique introduced in Chapter 2, Section 2.1. The commonly used Single Error Correction-Double Error Detection (SEC-DED) coding is applied, thus 7 redundant check bits are required for each register in the RF. The architecture of the SEC-DED-protected RF is shown in Figure 4.11. The redundant bits are added into each register, the write and read logic are amended to include the ECC encoding and decoding circuity. The principles of the ECC encoder and decoder can be found in Chapter 2, Section 2.1.2.1.

4.4.3 Statistical Analysis of Failure Rates for RFs

The soft error failure rate model introduced in Chapter 3, Section 3.6 was adapted for modelling the failure rates of the implemented RFs under radiation hits. A failure of a register file is defined as the corruption of single or multiple bits in one of its registers as results of particle strikes. This section calculates the probability of the occurrence of SBUs and MBUs of a single register in the RF.

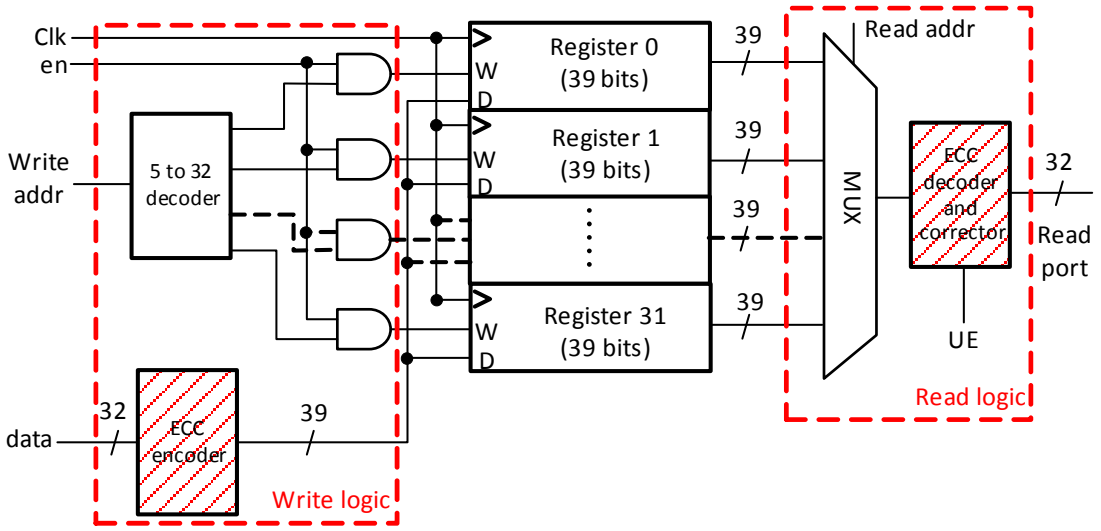

Figure 4.11: The SEC-DED-protected RF.

4.4.3.1 SET Failure Rate Model for the Register File

The write operation of the RF can cause single or multiple SETs to be sampled by multiple bits and result in SBUs or MBUs in the target register. The probability of SET-induced SBUs and MBUs in a register can be derived from the SET failure rate of a single flip flop proposed in Chapter 3, Section 3.6.1. Consider a register which has n flip-flops affected by SETs, the probability of these n SETs corrupting m bits in the affected flip-flops is given as below:

$$Fr(m) = C_m^n \times Fr^m \times (1 - Fr)^{(n-m)} \tag{4.2}$$

where $0 \leq m \leq n$, Fr^m denotes the probability of m errors calculated as the product of the failure probability of m flip-flops. C_m^n is the combinatorial function which calculates the number of cases that m errors occur when the register is affected by n SETs.

It should be noted that for deriving the failure of a conventional RF or an ECC-protected RF [2], Fr should be calculated using Equation 3.15 in Section 3.6.1 of Chapter 3. For deriving the failure of the self-checking register-protected RF, Fr should be replaced by Fr', which can be calculated using Equation 3.18 in Section 3.6.1 of Chapter 3.

4.4.3.2 SEU Failure Rate Model for Register File

The probability of an SEU-induced failure for a conventional RF is 100%. This is because one or more bits in one of the registers will be corrupted. All the single-bit SEUs or

[2]The ECC does not have SET-tolerant capability, thus the failure rate of an ECC-protected RF is the same as that of the original RF.

multiple-bit SEUs in the self-checking register-protected RF can be corrected on the fly by the incorporated SETTOFF2s. As the results showed in Chapter 3, Section 3.7.5.3, the SEU failure rates of any single SETTOFF2 is zero. Therefore the SEU failure rate of the self-checking register-protected RF is also 0. The SEC-DED-protected RF can correct SBUs and thus reduce the SBU failure rate to 0. It also detects double-bit errors. However, SEUs corrupting more than 2 bits of a register will escape detection and produce a failure rate of 100%

4.4.4 Experimental Setups and Comparative Evaluation Results for the Radiation Hardened RFs

4.4.4.1 Experimental Setups

The self-checking register-protected RF and the SEC-DED-protected RF are compared in terms of their reliability and error-tolerance overheads. The SPICE netlists of the 3 RFs are constructed for evaluation. The 3 RFs are all 32-bit and each contains 32 registers, 1 read port and 1 write port. The power consumption and propagation delay of the RFs are derived from SPICE simulations, based on the threshold and transition time defined in the library (see Chapter 3, Section 3.7.4). The area of the 3 RFs are compared by counting the number of transistors required for constructing the RFs. The reliability of the 3 RFs are analysed using the failure rate model described in Section 4.4.3, and the comparative results are illustrated in Section 4.4.4.2. The results for the implementation overheads of 3 RFs are presented in Section 4.4.5.

4.4.4.2 SET and SEU Failure Rate Results

A 1GHz symmetric clock is applied for analysing the SET-tolerance capability of the RF using the failure rate model. According to the results derived in Chapter 3, Section 3.7.5.2, for a 1GHz symmetric clock, the average SET failure rate of a single SETTOFF2, $Fr' = 4\%$, and the average SET failure rate of a conventional flip-flop, $Fr = 45\%$ in 65nm technology. Based on the Fr and Fr', the SET failure rates of the conventional RF and the self-checking register-protected RF are derived from Equation (4.2). According to the results of the occurrence of multiple SETs reported in [56], the number of the SETs presented at the input of the RFs, n is set to 5 for this evaluation.

Table 4.3 shows the failure rates of different number of SET-induced MBUs for the original RF, the self-checking RF, and the SEC-DED-protected RF. The probability that the original RF is not corrupted by the 5 SETs occurring at its input is only 5%; while 82% of the multiple SETs do not induce an error in the self-checking RF. The self-checking RF also significantly reduced the failure rates of different numbers of SET-induced MBUs. Notice that ECC does not have an SET-tolerance capability.

Table 4.3: The SET-induced MBU failure rates for the self-checking register-protected RF, the original RF, and the SEC-DED-protected RF.

No. of SET-induced MBUs	Nil	1 bit	2 bits	3 bits	4 bits	5 bits
Self-checking register-protected RF	82%	17%	1.4%	0.06%	1E-3 %	1E-5%
Original / SEC-DED-protected RF	5%	21%	34%	28%	11%	18%

Table 4.4: The SEU-induced MBU failure rates for the self-checking register-protected RF, the original RF, and the SEC-DED-protected RF.

No. of SEU-induced MBUs	1 bit	2 bits	more than 2 bits
Original RF	100%	100%	100%
SEC-DED-protected RF	0%	detected	100%
Self-checking RF	0%	0%	0%

Table 4.5: Implementation overheads of the robust RFs in 65nm technology.

	Area overhead [3]	Power overhead	Performance overhead
Proposed RF	78%	46%	15.2%
ECC-protected RF	30%	90%	extra cycle

SET pulses originated from the preceding logic can propagate through the encoder that generates the ECC bits. The encoder itself can also produce SETs under radiation hits. Therefore the SET failure rate of an ECC-protected RF is at least the same as that of the conventional RF.

The failure rates of the SEU-induced MBUs for the three implemented RFs are shown in Table 4.4. As mentioned in Section 4.4.3.2, the self-checking register-protected RF has the best MEU-tolerance capability.

4.4.5 Implementation Overheads

Table 4.5 summarises the implementation overheads of the self-checking RF and the SEC-DED-protected RF in 65nm technology. With a 10% activity rate for each bit, the SEC-DED-protected RF incurs an average 90% power overhead during a write cycle, while the proposed register consumes an average of 46% extra power. The self-checking RF incurs a delay overhead of 15.2% due to the extra loads added for error-tolerance. The delay overhead of the SEC-DED-protected RF is big due to the large decoding block at the read port, therefore an extra cycle may be required to reload the register for error correction. It should be noted that these results are derived from an RF with only 1

[3]The area overhead is estimated based on the number of transistors.

read port. The area and power overhead of the ECC technique will increase significantly for protecting an RF with multiple read ports. This is because multiple ECC decoders and correction logic are required for multiple read ports.

4.5 Concluding Remarks

A common drawback of most previous radiation hardened techniques is the lack of self-checking capabilities. Radiation hardening typically requires hardware redundancies which can also be affected by transient faults, and hence introduce additional vulnerabilities to the circuit. Most previous techniques only use redundancies to mitigate the errors occurring in the original circuitry, but cannot address the errors occurring in the redundancies. The severity of this problem depends on the probability of the errors arising from the redundancies, and the types of errors the redundancies can produce. The ECC decoder and encoder, for instance, can produce SETs which can turn into soft errors if captured. The widely used C-elements can produce SEUs which could corrupt the cell directly.

This chapter proposed a self-checking radiation hardened technique based on the SETTOFF architecture proposed in Chapter 3. The new technique further improves the reliability of the SETTOFF technique by adding a self-checking capability to mitigate the errors occurring in the redundancies. The self-checking capability is achieved by using a self-checker, which can be shared by multiple bits in a register architecture constructed by SETTOFFs. The sharing of the self-checker minimises the overheads that the self-checking capability introduced. The evaluation results show that compared to the original SETTOFF architecture, the increase of the area, power consumption, and delay in the self-checking architecture is insignificant. However, the self-checking technique provides a noticeably better error-tolerance capability since the SETs and SEUs in both the original circuitry and the redundancies can be mitigated.

In addition, the register file is one of the most vulnerable blocks of a microprocessor. So in order to validate the efficiencies of the proposed self-checking technique, it is implemented for protecting the register file. The reliability and the error-tolerance overheads of the register file protected by the proposed technique are then compared with a typical ECC-protected register file. The failure rate model, proposed in Chapter 3, Section 3, is adapted to statistically analyse the reliability of the register-level implementations. The results show that the proposed self-checking technique produces much lower SBU and MBU failure rates caused by the SETs and SEUs affecting the RF. Meanwhile, the proposed technique requires a bigger area, but much smaller power consumption and delay overheads than the ECC technique for protecting the RF.

Chapter 5

The Self-Checking Radiation Hardened OpenRISC Pipeline Design

As reviewed in Chapter 2, Section 2.5, a number of pipeline protection mechanisms have been proposed based on radiation hardening techniques, error coding techniques, or duplication. However, due to the complexity of the processor pipeline and the limited error-tolerance capabilities that these techniques can provide, few of the previous pipeline protection mechanisms can achieve a complete protection for the entire pipeline. Some of them, such as the Razor and SEM/STEM techniques can only protect the pipeline registers, while others, such as the Parshield and RRC, are only suitable for protecting the RF within the pipeline.

Since the SETTOFF-based self-checking techniques proposed in Chapter 3 and Chapter 4 can provide efficient error-tolerance capabilities for both SETs and SEUs, they can be applied to a microprocessor to achieve error-tolerance. This chapter proposes a complete and cost-efficient pipeline protection mechanism based on the self-checking architectures. The radiation hardened pipeline is achieved by incorporating the SETTOFF-based self-checking cells into the sequential cells constructing the pipeline. A replay recovery mechanism is also developed at the architectural level to interact with the self-checking cells to recover the corresponding detected errors. The proposed pipeline protection technique is implemented in an OpenRISC microprocessor in 65nm technology. A gate-level transient fault injection and analysis technique is developed for evaluating the error-tolerance capability of the proposed hardened pipeline design.

This chapter is organised as follows: Section 5.1 introduces the OpenRISC processor used for implementing the proposed pipeline protection mechanism. Section 5.2 proposes the gate-level transient fault injection and analysis technique, which is used for analysing the soft error vulnerability of the unprotected OpenRISC processor. Based

on the analysis results, the self-checking radiation hardened pipeline design is proposed in Section 5.3. The experimental methodology for implementing the processor is described in Section 5.4, and the evaluation results are presented in Section 5.5. Finally, the chapter is concluded in Section 5.6.

5.1 The OpenRISC 1200 Microprocessor

OpenRISC 1000 is a free and open architecture for a family of RISC microprocessor cores. It is designed with the emphasis on performance, simplicity, low power requirements, and scalability [84]. OpenRISC 1200 (OR1200) is a synthesizable RTL implementation of the OpenRISC 1000 architecture [85]. It is a 32-bit scalar RISC machine implemented with the ORBIS32 [1] instruction set architecture and Harvard micro-architecture. OR1200 has 5 stage integer pipeline, 53 distinct instruction op-codes and three addressing modes: immediate, displacement and pc-relative. The general architecture of the OR1200 IP core is shown in Figure 5.1. The CPU core is the central block of the OR1200 processor. The IP core has a separate level-1 data cache and instruction cache, which are connected to their separate Memory Management Units (IMMU and DMMU). Only the write-through policy is supported by the data cache. The IMMU and DMMU communicate with the on chip and off chip memories through WISHBONE interfaces (IWB and DWB). The IP core also consists of the following units:

- Power Management Unit (PMU): The OR1200 core has a sophisticated power management support to optimize the power consumption. The CPU core can operate in 4 different modes, (1) Slow mode, (2) Idle mode, (3) Doze mode, (4) Sleep mode, which dynamically activate or deactivate certain internal modules.
- Debug Unit: It supports basic debugging for OR1200. Advanced debug features such as the watchpoints and breakpoints are not supported.
- Tick Timer: The tick timer is clocked by the OpenRISC clock, and is used by the operating system for precisely measuring time and scheduling system tasks.
- Programmable Interrupt Controller (PIC): The PIC receives interrupts from external sources and forwards them to the CPU core as low or high priority.

5.1.1 OR1200 CPU Core

The OR1200 core consists of 13 modules whose abbreviations and actual names are listed in Table 5.1. The block diagram of the OR1200 core is shown in Figure 5.2, which

[1] This is the OpenRISC Basic Instruction Set with 32-bit wide instructions aligned on 32-bit boundaries in memory, and operating on 32-bit and 64-bit data. Refer to [84] for more details.

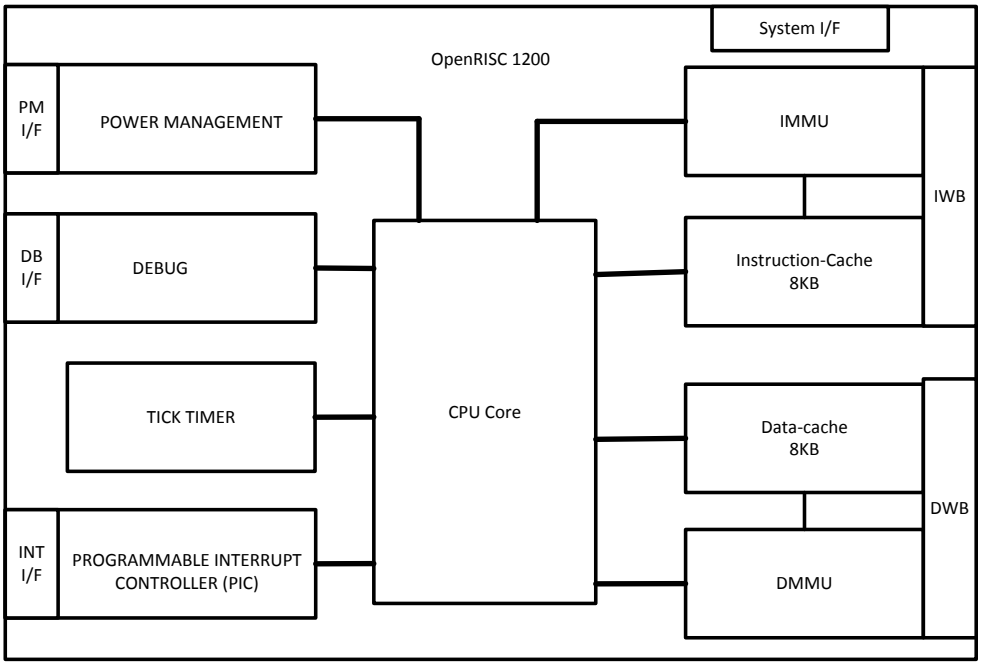

Figure 5.1: Architecture of the OpenRISC 1200 IP core [85].

Table 5.1: Abbreviations of the blocks inside CPU core

Module Abbreviation	Name
Genpc	Program counter generator
If	Instruction fetch unit
Ctrl	Control unit
RF	Register file
Operandmux	Operand multiplexer
SPR	Special purpose registers
ALU	Arithmetic logic unit
Except	Exception unit
Mult_Mac	Multiply accumulate unit
LSU	Load and store unit
Cfgr	Configuration Registers
Freeze	Freeze logic
Wbmux	Write back multiplexer

illustrates the main modules and their positions within the pipeline. Notice from the left hand side of Figure 5.2 that although OR1200 has 5 pipeline stages, the memory access stage (MEM) is optional, and is only invoked for a load or a store instruction to exchange data with the data cache (DC). Otherwise, the operation goes directly into the WB stage after the EX stage to write the execution results into the RF.

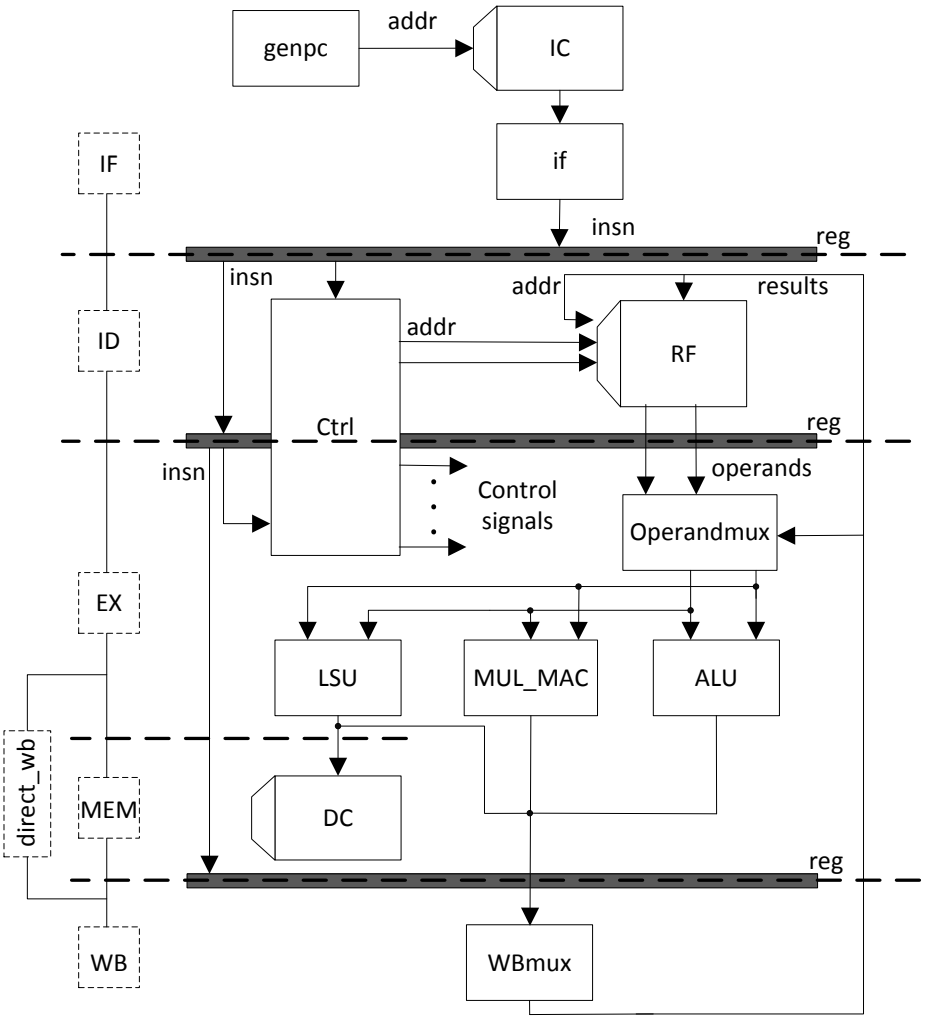

Figure 5.2: Block diagram of the OR1200 core [86].

The functionalities of some important modules inside the CPU core are briefly described as follows:

- *if*: The *if* module is connected to the instruction cache to fetch instructions. It also passes instructions to the *ctrl* module for decoding.
- *ctrl*: This is the main controller of the CPU core that conducts most of the instruction decoding processes. *ctrl* generates control signals, addresses, and immediate operands for the execution of the instructions.
- *RF*: The register file that contains 32 general purpose registers.

- Mul_Mac: This module is responsible for executing the multiplication operations.
- $cfgr$: This module controls the configurations of the CPU core through special purpose registers
- $except$: This module handles different kinds of exception conditions, such as external interrupt requests, internal errors, system calls, arithmetic overflow, and memory access conditions.
- $freeze$: This module generates signals to freeze the pipeline stage when needed.
- lsu: This module communicates with the data cache to exchange data for a load or a store instruction.

5.1.2 OR1200 Pipeline Architecture and Operating Principle

In order to further explain the operating principle of the OR1200 pipeline, the register-level abstraction of the pipeline is extracted in Figure 5.3. All the combinational logic blocks are represented by white boxes, while the sequential blocks (including the flip-flops, registers, and the caches) are presented by orange boxes. The blue boxes in between each stage of the pipeline contains the pipeline registers that connects the adjacent pipeline stage. These pipeline registers store data, instructions, and control signals intermediately for the corresponding pipeline stage.

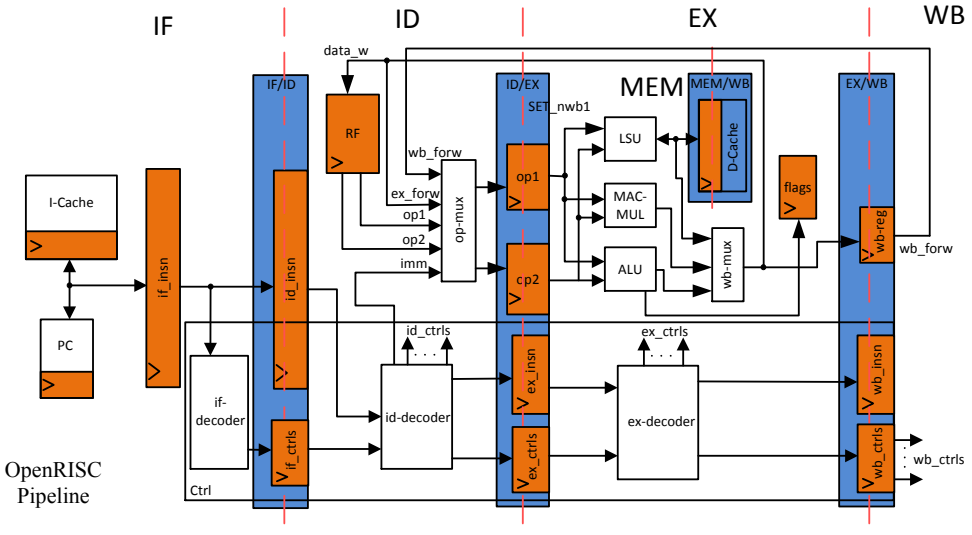

Figure 5.3: Register abstraction view of the OR1200 pipeline.

During the Instruction Fetch (IF) stage, instructions are fetched from the instruction cache, and are stored in the if_insn register. The instruction decoder in the IF stage, $if - decoder$, then decodes the instruction in if_insn to generate control signals to

be saved in if_ctrls flip-flops. The instruction in if_insn is also forwarded into the id_insn register which feed the instruction into the Instruction Decode (ID) stage. Most instruction decodings are committed during the ID stage. The instruction stored in the id_insn register is decoded by the decoder in the ID stage ($id-decoder$), which generates immediate operands, control signals, and addresses used for reading data from the RF. The operand multiplexer ($op-mux$) then selects 2 operands from the data read from the RF, or the immediate operands from the $id-decoder$, or the results forwarded from the EX or WB stages of the pipeline. Before execution in the EX stage, the 2 selected operands are stored in the 2 operand registers ($op1$ and $op2$). In the mean time, the instruction in the ID stage is forwarded to the ex_insn register, and the control signals decoded during the ID stage are stored in the corresponding control flip-flops (ex_ctrls). During the execution (EX) stage, the execution units (ALU and MAC-MUL) conduct specific operations based on the 2 operands and the controls generated by the decoder in the EX stage ($ex-decoder$). In the case of executing a load or a store instruction, the Load and Store Unit (LSU) in the memory access stage (MEM) is invoked for exchanging data with the data cache. The final execution results are then selected by the write back multiplexer (wb-mux), which writes the results into the RF and the flag registers at the beginning of the the WB cycle, using the rising clock edge. Meanwhile, the execution results are also written into the $wb-reg$ register, while the instruction and the control signals in the EX stage are forwarded into the wb_insn and wb_ctrls in the WB stage, respectively. Notice that the RF, the flag registers, the wb_insn, the wb_ctrls, and the wb_reg are all updated at the same time, which is at the beginning of the WB cycle.

The pipeline forwarding technique is applied in the OR1200 pipeline which can forward the execution results in the EX stage and the WB stage back into the ID stage (refer to the ex_forw and wb_forw signals). The EX-forwarding technique feeds the output of the $wb-mux$ back to the $op-mux$. This improves the pipeline efficiency and solves the data hazard problem when the instruction in the EX stage needs the execution results from the previous instruction as the execution operands. The WB-forwarding technique feeds the execution results from the WB stage back into the MEM stage, so that the following store instruction can save the results into the $D-Cache$.

5.2 Soft Error Vulnerability Analysis for the OpenRISC Pipeline

The basic idea of the proposed pipeline protection technique is to protect the sequential cells (such as registers and flip-flops) within the pipeline using the circuit-level and register-level radiation-hardening approaches described in Chapter 3 and Chapter 4. Because only the SETs captured by the sequential elements can propagate through the pipeline, the proposed circuit-level and register-level cells can combat both the SEUs occurring in the storage element, and the SETs occurring in the combinational logic when

they are captured. Therefore, if all the sequential cells are protected by the proposed radiation-hardening approaches, the proposed pipeline protection technique can actually protect against all types of soft errors occurring within the pipeline. However, such full protection can result in big error-tolerance overheads due to the large number of sequential cells inside the pipeline.

Mehdizadeh, et al. presented an analysis of the fault effects in the OpenRISC processor in [86]. The authors injected different types of faults (such as stuck-at faults and bit-flips) into the processor, and checked whether the execution results were corrupted by the faults. The analysis results show that different units in OR1200 present different vulnerabilities to the injected faults. The levels of vulnerability of different units are independent of the programs that the processor runs. The register file unit, for instance, is always more vulnerable than the WB-multiplexer unit regardless of the programs that the processor runs. Ebrahimi, et al. also presented an SER analysis of the OpenRISC processor implemented in 45nm technology [24]. The results show that 75% of the flip-flops within the microprocessor make negligible contributions to the overall system SER, and by protecting 20% of the flip-flops which are most vulnerable, the system SER caused by all the flip-flops can be reduced by 80%. The authors in [24] have also shown that the level of vulnerability of the flip-flops, again, has little dependency on different workloads.

All these observations suggested that not all the registers inside the pipeline have the same level of vulnerability to soft errors. The reasons for this are explained as follows:

1. Not all the soft errors occurring within the pipeline can corrupt the pipeline execution. For instance, a soft error occurring in a register that will not be read will not affect the normal pipeline operation.

2. Some cells inside the pipeline, such as the pipeline registers and RF, are more frequently accessed during the execution than other cells. Therefore, when corrupted by soft errors, they will have a higher probability to corrupt the pipeline operation compared to other less frequently accessed cells.

3. Not all the cells are equally important for the pipeline operation. For an instance, the control unit generating the control signals can easily corrupt the pipeline operation when itself corrupted. However, the errors occurring in the intermediate data may be masked during speculative operations.

Full protection can be inefficient since it may sacrifice unnecessary area and power to protect cells that are unlikely to contribute to the overall SER of the processor. In order to achieve an effective and cost-efficient protection, a system-level analysis of soft error effects on the OpenRISC processor were carried out. The analysis identified the soft error vulnerability of different registers inside the pipeline. The safety-critical ones

(i.e. the registers which are most vulnerable to soft errors) can then be selected for protection.

5.2.1 Transparent Fault Injection and Simulation in VHDL

A simulation-based transient fault injection and simulation technique was developed for achieving the soft error vulnerability analysis. The technique can inject different types of transient faults into different parts of the system during gate-level simulation. The simulation results were then collected for statistically analysing the soft effects in different parts of the system. The proposed transient fault injection and analysis technique was developed from the transparent fault injection and simulation technique proposed by Zwolinski in [87] and [88]. Before introducing the proposed analysis technique, an overview of the transparent fault injection and simulation technique is provided, as follows:

Simulation-based fault-injection techniques are typically realised by perturbing a circuit model and repeating the simulations. Conventionally, this is achieved by adding extra ports and control wires to the circuit under simulation, such that particular control signals can be applied to inject and activate particular faults in the circuit. However, such approaches have the following drawbacks: A large number of control wires may be needed for comprehensively evaluating a large system. This may require a radical modification of the circuit; it would also require a complex testbench to switch the control signals for activating the faults. In addition, the faults are difficult to uniquely identify, making the fault effect difficult to record.

The transparent fault injection and simulation technique is achieved by injecting faults into a VHDL description of the circuit gate model using shared variables. The shared variable is provided in the 1993 VHDL standard as an additional way of transferring values across modules other than generic and ports. For injecting faults into a circuit, the gate models that construct the circuit are modified to include non-activated pre-determined faults. All the pre-determined faults in all gates are contained in a linked list structure, with each element representing a particular fault in a gate. A shared variable is declared as a pointer, which moves along the linked list and controls the fault injection process by activating the pre-determined faults one by one during simulation. The technique does not require any extra controls wires, therefore during the fault simulation, the circuit netlist remains unchanged compared to the fault-free simulation. Only the VHDL descriptions of the gate models need to be changed to include the faults, and a testbench needs to be created for the fault simulation. A detailed example of applying the transparent fault injection and simulation technique is given below.

The first step is the declaration of a package for building the data structure during the fault injection process. Figure 5.4 shows the fault injection package which contains the following information:

fault_model : This is a record created for modelling each fault. The record contains the *faul_name*, 2 boolean flags (*simulating* and *detected*), and a pointer pointing to the next fault model. The *simulating* flag indicates whether the fault is being activated; while the *detected* flag indicates whether the fault is detected.

fault_ptr : This is a pointer pointing to the *fault_model*. It is declared using a VHDL **access** type.

first_fault : This is a shared variable that is a pointer, pointing to the first fault in a linked list. The shared variable is initialized to **null**. It can pass values to any other modules that use this package.

```
1  use std.textio.all; -- contains definition of line
2  package fault_inject is
3    type fault_model; -- incomplete type declaration
4    type fault_ptr is access fault_model;
5
6    type fault_model is
7      record
8        fault_name : line; -- line is access string
9        simulating : boolean;
10       detected   : boolean;
11       next_fault : fault_ptr;
12     end record fault_model;
13
14   shared variable first_fault : fault_ptr := null;
15 end package fault_inject;
```

Figure 5.4: The package declared for fault injection [87].

The package can then be used for including the pre-determined faults in the VHDL descriptions of the gate modules. Figure 5.5 shows an example of a two-input NAND-gate modified for including fault models using the package. Three stuck-at faults are included in the gate, which are the output Z stuck at 1, the input A stuck at 1, and the input B stuck at 1. The three local variables, z_sa1, a_sa1, and b_sa1, point to the three faults, respectively. At the beginning of the simulation, the variable z_sa1 is set to **null**, thus the data structure is created by lines 13 to 21 of the code in Figure 5.5. Initially, the *simulating* and *detected* flags are assigned to be false to allow a fault-free simulation. The *fault_name* is generated by the *instance_name* attribute appended with the name of the variable. The *instance_name* attribute creates the full name of the path from the top-level hierarchy of the system. The *fault_name* allows each fault to be uniquely identified.

It should be noted that no extra ports or control wires are added to the modified NAND-gate. Therefore, it can be used for modelling the included faults in any circuit that is

```
1  use IEEE.std_logic_1164.all;
2  use WORK.fault_inject.all;
3  entity nand2 is
4    port (z : out std_logic; a, b : in std_logic);
5  end entity nand2;
6
7  architecture inject_fault of nand2 is
8  begin
9    nn : process (a, b) is
10       variable z_sa1, a_sa1, b_sa1 : fault_ptr := null;
11   begin
12     if z_sa1 = null then
13       z_sa1 := new fault_model'(
14                   new string'(inject_fault'instance_name & z_sa1),
15                               false, false, first_fault);
16       first_fault := z_sa1;
17       a_sa1 := new fault_model'(
18                   new string'(inject_fault'instance_name & a_sa1),
19                               false, false, first_fault);
20       first_fault := a_sa1;
21       b_sa1 := new fault_model'(
22                   new string'(inject_fault'instance_name & b_sa1),
23                               false, false, first_fault);
24       first_fault := b_sa1;
25     end if;
26     if fault_sim.simulating(z_sa1) then -- z/1
27       z <= '1';
28     elsif fault_sim.simulating(a_sa1) then -- a/1
29       z <= not b;
30     elsif fault_sim.simulating(b_sa1) then -- b/1
31       z <= not a;
32     else -- fault-free
33       z <= a nand b;
34     end if;
35   end process nn;
36 end architecture inject_fault;
```

Figure 5.5: Two-input NAND gate with fault injection model [87].

constructed with this gate. Figure 5.6 shows a circuit that is constructed with 3 two-input NAND-gates modified with the fault injection model in Figure 5.5. The solid lines are the physical wires connecting the gates together. The data structure is created at the beginning of the simulation, where all the faults included in each gate are linked together in a linked list. A template of the fault simulation testbench is shown in Figure 5.7. During the simulation process, a fault free simulation is performed with certain test vectors to extract the correct circuit response. Afterwards, sequential fault simulations are carried out using the same test vectors. A local pointer (*head_ptr*) is created which moves along the linked list and activates one fault during each simulation. The fault simulation results are then compared with the fault-free simulation results to identify errors. The simulation process is achieved by the local pointer (*head_ptr*) controlling the shared variable (*first_ptr*) through the dash line in Figure 5.6.

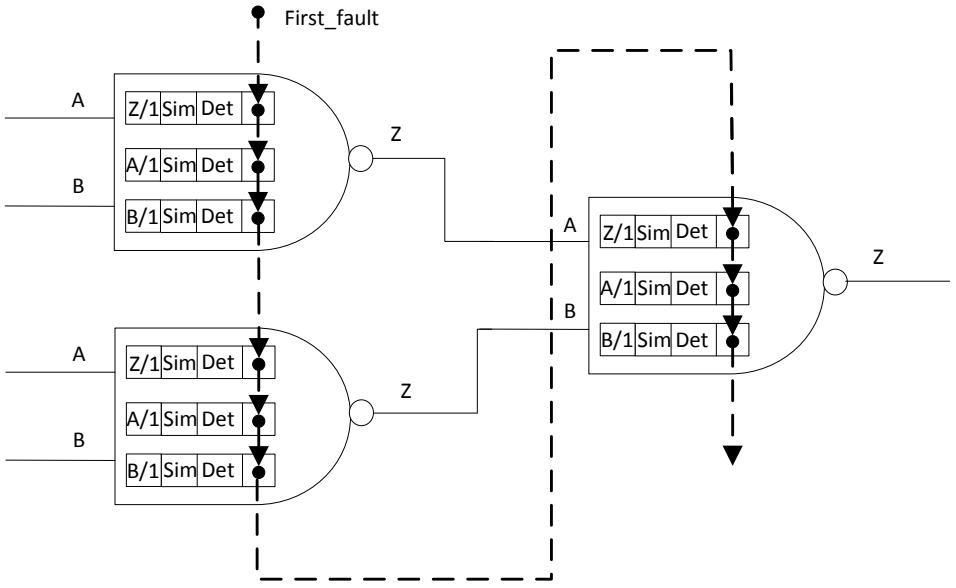

Figure 5.6: Date structure of the fault model [87].

```
1    sim : process is
2          variable head_ptr : fault_ptr;
3      begin
4        -- FAULT-FREE SIMULATION
5
6        -- apply test vectors
7        -- apply Xs
8
9        -- SEQUENTIAL FAULT SIMULATION
10       head_ptr := first_fault;
11       while head_ptr /= null loop
12          head_ptr.simulating := true;
13          -- apply test vectors
14          -- compare with fault free case and print differences
15          head_ptr.simulating := false;
16          -- apply Xs
17          -- move to next fault;
18          head_ptr := head_ptr.next_fault;
19       end loop;
20     end process sim;
```

Figure 5.7: Fault simulation testbench template [87].

5.2.2 The Transient Fault Injection and Analysis Technique

A system-level transient fault injection and analysis technique was developed from the transparent fault injection method described in Section 5.2.1. The proposed technique can inject radiation-induced transient faults into a system and collect the system response data via gate-level simulations. Similarly to the method described in Section 5.2.1, the proposed technique uses the fault model package described in Figure 5.4

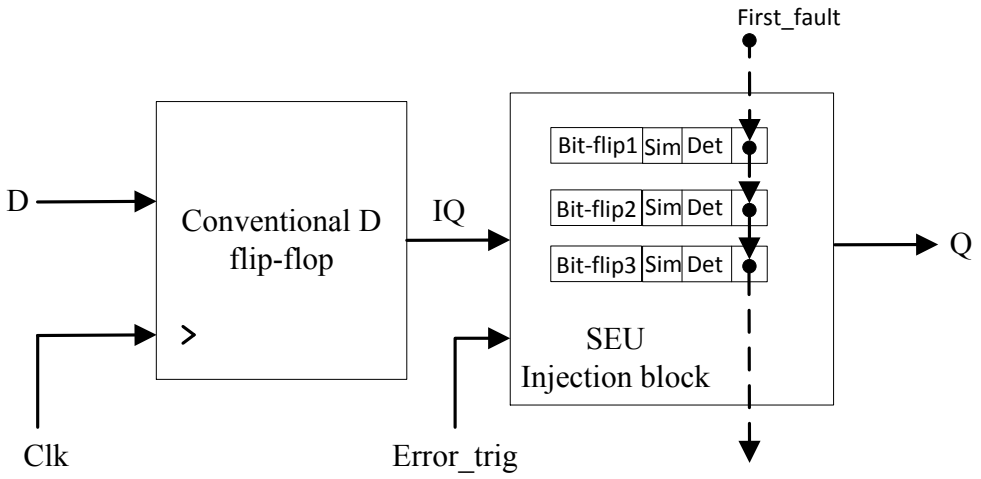

Figure 5.8: The modified D flip-flop for SEU injection.

to include pre-determined faults into the gates. The linked list data structure is also used for the simulation. However, the proposed technique models transient faults (i.e. the SETs in combinational blocks, and the SEUs in sequential blocks) rather than stuck-at faults. The details of the SET and SEU modelling methods are described separately, as follows:

5.2.2.1 SEU Injection into Sequential Gates

The transient fault injection technique models SEUs as transient bit-flip errors corrupting the outputs of sequential gates. Figure 5.8 shows an example of how the bit-flip errors are modelled and included in a conventional D-type flip-flop. Since it is a sequential gate, the injection method is different from the one in Section 5.2.1. An SEU-injection block was added to the output of the conventional flip-flop such that the original output becomes an internal signal IQ. The SEU-injection block propagates IQ to the actual output (Q) of the modified flip-flop during fault-free simulation. Three bit-flip error models (bit_flip1, bit_flip2, and bit_flip3) are included as a linked list structure and are pointed to by the shared variable $first\_fault$. An $Error\_trig$ signal is added for controlling the activation of the errors together with the *simulation* Boolean flag in the fault model. The state of IQ will be flipped to Q when any of the SEUs are activated.

Figure 5.9 shows the VHDL description of the modified flip-flop shown in Figure 5.8. It should be noted that an extra port for the $Error\_trig$ signal was added to the flip-flop. The CLK, $Error\_trig$, and IQ are all in the sensitivity list of the SEU-injection block. A rising edge on CLK or any transitions on IQ mean the state stored in the

```
1  Library IEEE; Use IEEE.STD_LOGIC_1164.all;Use work.fault_inject.all;
2  entity DFF_SEUinject is
3          port(
4                  Q               :       out     STD_LOGIC ;
5                  D               :       in      STD_LOGIC ;
6                  CLK             :       in      STD_LOGIC ;
7                  Error_trig      :       in      STD_LOGIC );
8  end DFF_SEUinject;
9
10 architecture VHDL_FUNCT of DFF_SEUinject is
11         SIGNAL  IQ      : STD_LOGIC;
12 begin
13
14 -- Conventional D flip-flop
15 PIQ : Process (CLK)
16     begin
17       -- Description of the conventional D flip-flop
18     end Process;
19
20 -- SEU-injection block
21      nn: Process (CLK, Error_trig, IQ) is
22
23    variable bit_flip1, bit_flip2, bit_flip3: fault_ptr := null;
24     begin
25      if bit_flip1 = null then
26        bit_flip1 := new fault_model'(
27                 new string'(VHDL_FUNCT'instance_name & bit_flip1),
28                             false, false, first_fault);
29        first_fault := bit_flip1;
30
31      -- create data structure
32
33      end if;
34
35      if (bit_flip1.simulating) then -- SEUs injection
36         If (Error_trig'event) Then -- activate bit-flip error via Error_trig
37           Q <=  to_X01(not(IQ)) ;
38         elsif (rising_edge(CLK) or (IQ'event)) Then -- detect the update of ff
39           Q <=  to_X01(IQ) ;   -- bit-flip error overwritten by new input
40         end if;
41
42      -- sequential fault simulations
43
44      elsif (rising_edge(CLK) or (IQ'event)) Then  -- fault-free simulation
45          Q <= to_X01(IQ)  ;
46      end if;
47          end process nn;
48
49 end VHDL_FUNCT;
```

Figure 5.9: The VHDL description of the modified D flip-flop for SEU injection.

flip-flop is updated by the new input value D. Therefore in such cases, IQ will pass its value to Q when no errors are activated (see line 44-46 of Figure 5.9). During the fault simulation, a pointer moves along the linked list that is constructed by the bit-flip error models. The *simulation* flag in each fault model will be asserted one by one during the simulation process. However, the bit-flip errors are not activated until any switching of the *Error_trig* signal is detected. Each switching of the *Error_trig* signal will activate the bit-flip error model, which has the asserted *simulation* flag, once. The activated bit-flip error will not be recovered until the flip-flop is updated by the new input value, when the bit-flip error will be overwritten (see line 35-40 of Figure 5.9).

```
1          head_ptr := first_fault;
2          while head_ptr /= null loop
3             head_ptr.simulating := true;
4             -- apply test vectors
5             wait for inject_time;  -- this defines the time the SEU occurs.
6             Error_trig <= '1'  -- the switching of the Error_trig allows the
       injection of a transient bit-flip
7             -- compare with fault free case and print differences
8             head_ptr.simulating := false;
9             -- apply Xs
10            -- move to next fault;
11            head_ptr := head_ptr.next_fault;
12         end loop;
13       end process sim;
```

Figure 5.10: Part of SEU simulation testbench template.

Figure 5.10 shows a testbench for simulating the SEU injection. Only the code for fault simulations is shown, other codes are omitted as they are similar to the ones in Figure 5.7. The *simulation* flag of a particular fault is asserted during each entire fault simulation. The *Error_trig* signal is initialized to '0', and is asserted after the *injection_time*, which is a random number that allows injecting the SEU at a random time during the program runtime. The *Error_trig* can also be switched multiple times to allow multiple SEU injections during the simulation of a single fault model.

5.2.2.2 SET Injection into Combinational Gates

The SETs are modelled as the transient voltage pulses occurring at the output combinational gates. However, a system (such as the OpenRISC microprocessor) typically consists of a significantly larger number of combinational gates than sequential gates. Therefore, a large number of sequential fault simulations would be required to carry out the SET analysis and identify the vulnerability of different combinational blocks. This is inefficient and impractical. By observing that only the SETs captured by the sequential gates can affect the system and thus need to be considered, other SETs can hence be ignored to save simulation time. The proposed SET-injection technique therefore only injects SET pulses at the input of the sequential gates following the combinational gates. This maximises the probability of the injected pulses being captured. The SET vulnerability of the combinational gates can then be identified by the vulnerability of their following sequential gates.

Figure 5.11 shows an example of how the SETs are injected into the combinational gates preceding a D-type flip-flop. An SET-injection block is added at the input of the flip-flop such that the original input becomes an internal node, ID. The actual input of the modified flip-flop, D, is the input of the SET-injection block. The SET-injection block contains two modified inverters and each includes 2 SET fault modules defined by the package in Figure 5.4. Similar to the SEU-injection method, an *Error_trig* signal is added for controlling the activation of the SETs together with the *simulating* flag.

All the fault models are constructed in a linked list data structure. Figure 5.12 shows the example code for the modified inverter. Parts of the testbench for simulating the SETs are shown in Figure 5.13. For simulating a particular pre-determined SET, the *simulating* flag is asserted throughout the simulation. However, the particular SET is only activated by the *Error_trig* signal which is asserted after the *injection_time*. The activated SET will propagate the input of the inverter to its output and generate a voltage-transition. After the *pulse_width* time, the *Error_trig* signal will be assigned to 0, such that the inverter will operate normally. The voltage-transition will hence be recovered and the voltage glitch (i.e. the SET pulse) with the width equal to the *pulse_width* will be formed at the output of the inverter. The *injection_time* and *pulse_width* variables can be randomly selected to allow injecting SETs with random widths at a random time during the program runtime.

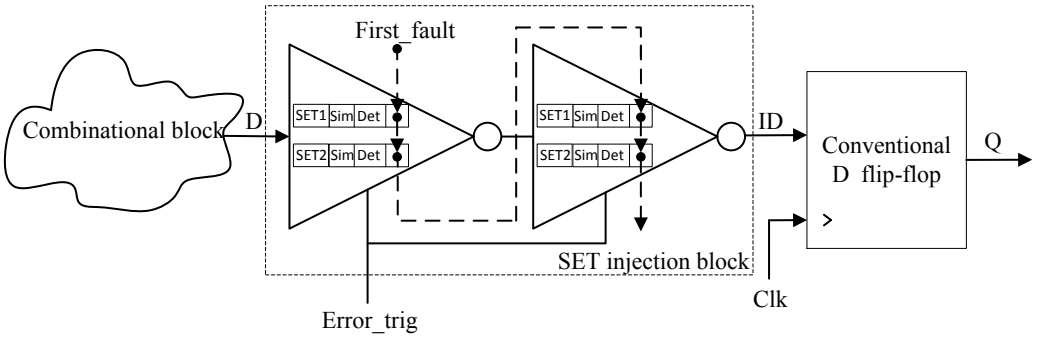

Figure 5.11: The modification of a D flip-flop for SET injection.

It should be noted that for both the SET and SEU injection methods, an extra signal, *Error_trig*, is added in the gate to be simulated. This is essential because injecting radiation-induced transient faults is different from injecting stuck-at faults. The transient faults only manifest themselves for a period of time, and will be recovered automatically. This requires the fault model to be activated and also deactivated at certain times during the simulation. The VHDL **Process** for fault injection needs to be sensitive to a signal such that it can be evaluated during the simulation to control the activation of the fault models. For both SET and SEU injection methods, only one global control signal *Error_trig* is required. The modification of the circuit netlist is relatively easy.

5.2.3 Vulnerability Analysis Results for the OpenRISC Pipeline

The proposed transient fault injection technique is used for analysing the soft error vulnerability of the OpenRISC pipeline. The RTL description of the processor is synthesized into a gate-level description using the STMicroelectronics 65nm technology. Since OpenRISC is a flip-flop based architecture, all the flip-flop cells in the technology library that construct the processor are modified, based on the SEU and SET injection

```
Library IEEE; Use IEEE.STD_LOGIC_1164.all;
-- entity declaration --
entity INV_SET is
  port(
    Z    : out STD_LOGIC ;
    A    : in  STD_LOGIC;
   Error_trig   :   in  STD_LOGIC);
end INV_SET ;
-- architecture body --
architecture VHDL_FUNCT of INV_SET  is
begin
  nn : process (A, Error_trig) is
      variable SET1, SET2 : fault_ptr := null;
  begin
    if SET1 = null then

    -- construct data structure

    end if;

       if SET1.simulating then
         if (Error_trig = '1') Then -- activate SET1
           z <= a;
         else
           z <= not a;
         end if;
       elsif SET2.simulating then
         if (Error_trig = '1') Then -- activate SET2
           z <= a;
         else
           z <= not a;
         end if;
       else    -- fault-free
         z <= not a;
       end if;

  end process nn;
end VHDL_FUNCT;
```

Figure 5.12: SET injection block.

```
        head_ptr := first_fault;
        while head_ptr /= null loop
           head_ptr.simulating := true;
           -- apply test vectors
           wait for inject_time;  -- this defines the time the SET occurs.
           Error_trig <= '1'  -- asserting the Error_trig activate the SET.
           wait for SET_width; -- this defines the width of the SET pulse.
           Error_trig <= '0'  -- assigning the Error_trig to 0 de-activate the
     SET.
           -- compare with fault free case and print differences
           head_ptr.simulating := false;
           -- apply Xs
           -- move to next fault;
           head_ptr := head_ptr.next_fault;
        end loop;
      end process sim;
```

Figure 5.13: Part of SET simulation testbench template.

techniques described in Section 5.2.2.1, and Section 5.2.2.2. We define a visible soft error as a corruption of the final program execution results stored in the data memory. The transient faults that are masked and do not affect the outcome of the programs will

be ignored. The analysis is carried out on the ORPSoc platform, which provides the smallest-possible reference system for testing the OpenRISC processor [89].

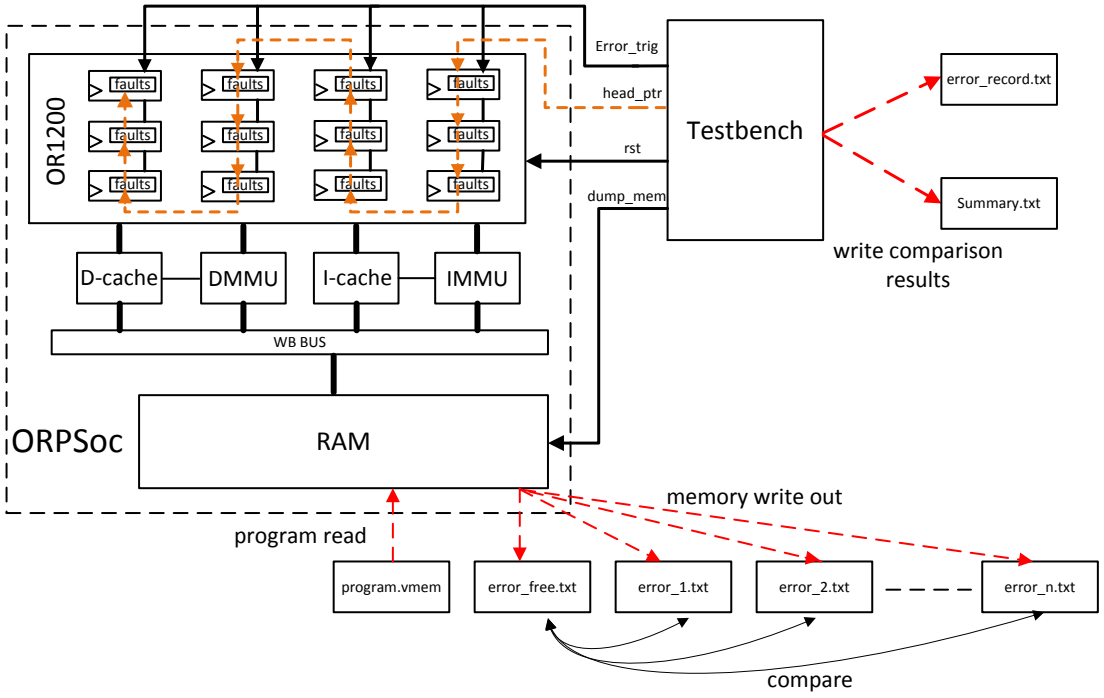

Figure 5.14: The soft error effect analysis model based on ORPSoc platform.

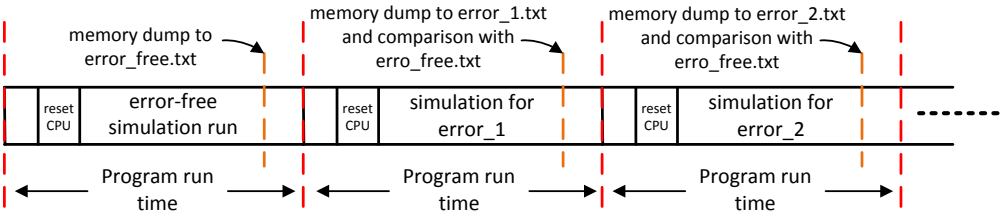

Figure 5.15: The timing diagram for soft error analysis simulation.

Figure 5.14 shows the block diagram of the soft error vulnerability analysis model based on the ORPSoc platform. The timing diagram of the soft error analysis simulation is shown in Figure 5.15. During the simulation, a C program is compiled into machine code (*program.vmem*) using the OpenRISC tool chain. The machine code is then loaded into the memory. A fault-free simulation is run first, and the correct execution results stored in the data memory are written out to a txt file (*error_free.txt*). Afterwards, the processor is reset and the same program is re-executed with one single fault (either an SET or an SEU) activated at a random time instance, each time re-running the program. The execution results for the transient fault simulations are also written out, and are

then compared with the results from the error-free simulation. If any inconsistencies in the results are detected, the particular transient fault that caused the inconsistency will be recorded in *error_record.txt*. After all the pre-determined transient faults in all the flip-flops within the pipeline are simulated, a summary detailing the total number of corruptions in the data memory and the total number of the injected faults is written out to *summary.txt*.

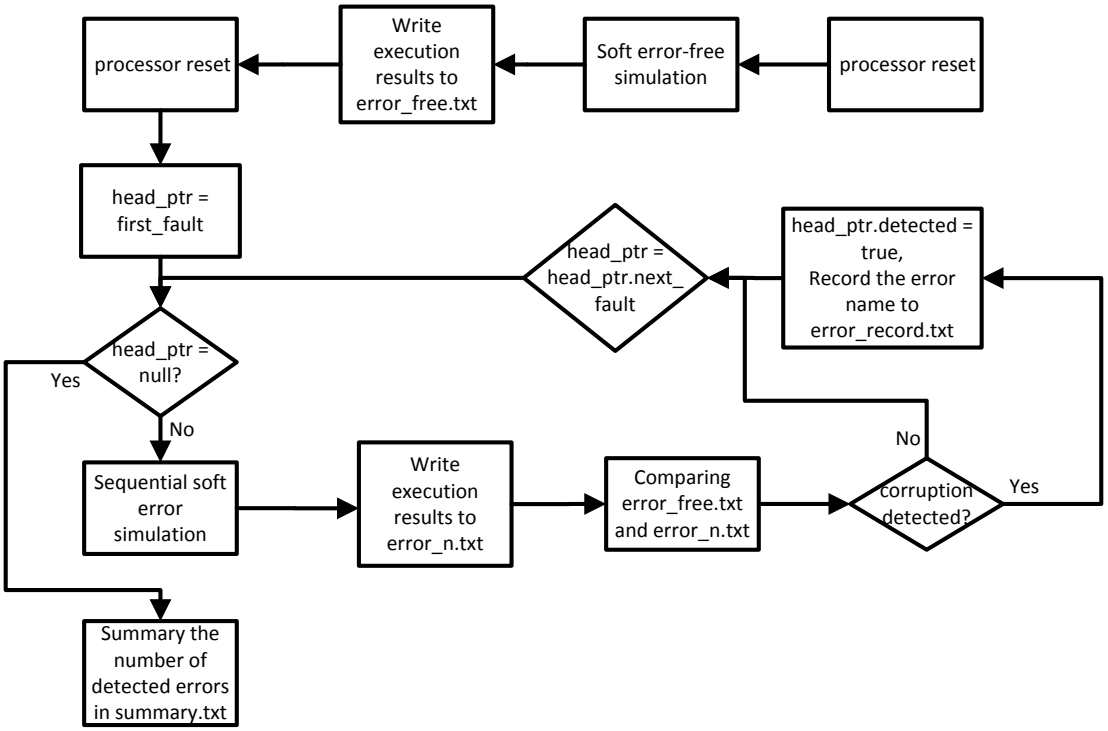

Figure 5.16: The soft error effect analysis simulation flow.

VHDL file operations are used in the testbench for extracting and writing out the analysis results during the simulations. Figure 5.16 shows simulation flow realised by the testbench, the code for which can be found in Appendix A. An example of the file operation code that compares *error_free.txt* and *error.txt* is shown in Figure 5.17. The data in the two files are compared line by line, and the soft errors are recorded in *error_record.txt* when any inconsistencies are detected. Part of the analysis results in *error_record.txt* are shown in Figure 5.18. The error record consists of the names of the injected faults, the numbers of the errors, the corrupted memory content and the expected memory content. An example for the summary of the analysis is shown in Figure 5.19.

Two programs, quicksort, which sorts an array of integers, and tak, which is a recursive function, are used for analysing the soft error effects on OpenRISC. Three soft errors,

```
        variable FFree, FInject : bit_vector(31 downto 0);
        file_open (memory, error_free.txt, read_mode);
        file_open (fault1, error.txt, read_mode);
      while not endfile(error_free) loop
        readline(error_free, ILine1);
        read(ILine1, FFree);
        readline(error, ILine2);
        read(ILine2, FInject);
        if FFree /= FInject then
          head_ptr.detected := true;
          fname := new string'(head_ptr.fault_name.all);
          writeline(error_record, fname);
          write(OLine, string'(Fault #), left, 0);
          write(OLine, fc, left, 5);
          write(OLine, string'(Detected by mem:), left, 0);
          write(OLine, FInject, right, 9);
          write(OLine, string'( expected: ), left, 0);
          write(OLine, FFree, right, 9);
          writeline(faults, OLine);
        end if;
       end loop;
        file_close(error_free.txt);
        file_close(error.txt);
```

Figure 5.17: Part of the analysis results in error_record.txt.

```
:test_orpsoc(fileio):s1:or1200_top0:or1200_cpu:or1200_genpc:\pcreg_default_reg
    [15] @hs65_ll_dfprqnx9(vhdl_funct)bit_flip2
Fault #10    Detected by mem:00000000000000000000000000000011 expected:
    00000000000000000000000000001001
:test_orpsoc(fileio):s1:or1200_top0:or1200_cpu:or1200_genpc:\pcreg_default_reg
    [15] @hs65_ll_dfprqnx9(vhdl_funct)bit_flip1
Fault #11    Detected by mem:00000000000000010011001100110011 expected:
    00000000000000010011100100111001
:test_orpsoc(fileio):s1:or1200_top0:or1200_cpu:or1200_genpc:\pcreg_default_reg
    [20] @hs65_ll_dfprqnx9(vhdl_funct)bit_flip
Fault #27    Detected by mem:00000000000000000000000000000000 expected:
    00000000000000000000000000001010
:test_orpsoc(fileio):s1:or1200_top0:or1200_cpu:or1200_genpc:\pcreg_default_reg
    [26] @hs65_ll_dfprqnx9(vhdl_funct)bit_flip1
Fault #44    Detected by mem:00000000000000000010010110110000 expected:
    00000000000000000010010111101000
:test_orpsoc(fileio):s1:or1200_top0:or1200_cpu:or1200_genpc:\pcreg_default_reg
    [7] @hs65_ll_dfpsqnx9(vhdl_funct)bit_flip
Fault #78    Detected by mem:00000000000000000000000000000000 expected:
    00000000000000000000000001010011
```

Figure 5.18: Part of the analysis results in error_record.txt.

```
Fault Cover:
      90 faults,       9 detected
```

Figure 5.19: An example of the analysis results in summary.txt.

including one SET and two SEUs, are injected into each flip-flop inside the pipeline of the processor. This is based on the assumption that the cell area that each bit occupies is the same, therefore the probabilities that each bit is struck by radiation particles are identical. The widths of the injected SETs are set as random numbers between 400ps to 600ps. This is in agreement with the SET distribution results for 65nm technology presented in [79].

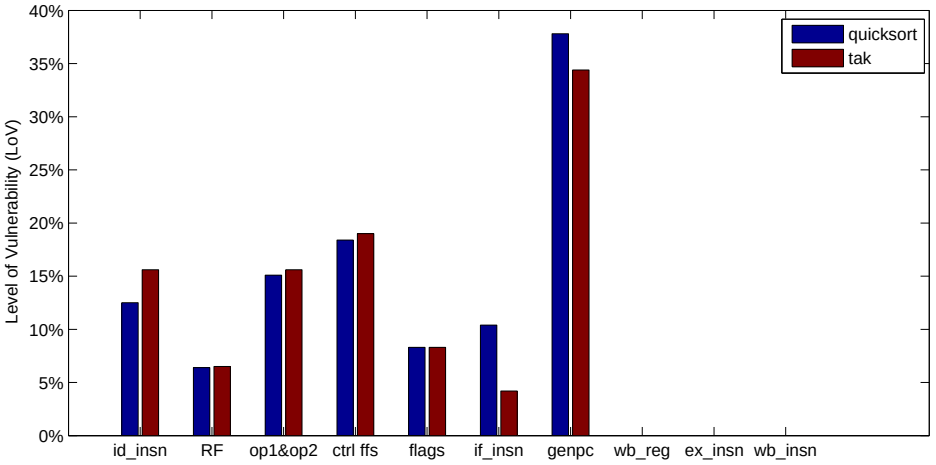

Figure 5.20: The soft error vulnerability analysis results for OpenRISC processor.

The Level of Vulnerability (LoV) is defined in Equation 5.1, which is equal to the number of transient faults that caused corruptions in the execution results divided by the total number of faults injected. According to Equation 5.1, the LoVs of different flip-flops and registers can be identified. The higher LoV produced by a register or a flip-flop indicates its higher vulnerability to transient faults.

$$LoV = \frac{\text{No. of transient faults that caused corruptions of program results}}{\text{No. of injected transient faults}} \tag{5.1}$$

The LoVs produced by different registers and flip-flops in the processor during the simulation of both of the programs are shown in Figure 5.20. The PC register produces the highest LoV in the simulation of both programs. A total of 90 transient faults are injected into the PC register during each program simulation, and 31 (37.8%) and 34(34.4%) of them caused corruptions in the execution results for the quicksort and tak program simulations, respectively. This is because the faults occurring in the PC can easily corrupt the pipeline execution by fetching erroneous instructions. The flip-flops that store the control signals ($ctrl\ ffs$) generated by the decoders in each stage of the pipeline presents the second highest LoV. This is because the control signals are also critical for correction pipeline execution. Other registers that present relatively high LoVs are: the registers that store the instruction for the ID stage (id_insn) and IF stage (if_insn); the registers that store the operand for the execution stage ($op1$ and $op2$); the RF, and the flag registers.

Notice that the registers that store the instructions in the EX stage (ex_insn) and the WB stage (wb_insn) produced 0% LoV despite the injection of 198 transient faults in both simulations. The reason for this is that most of the controls signals are decoded

in the IF stage and ID stage by using the instructions stored in if_insn and id_insn. The control signals are then saved in the ex_ctrls, which are the flip-flops that store the control signals in the EX stage (see Figure 5.3). The ex_insn register is very rarely used for the instruction decoding in the EX stage, hence the transient faults occurring in the ex_insn can hardly affect the pipeline execution. Similarly, the wb_insn is not used for the decoding process. The control signals for the WB stage stored in wb_ctrls flip-flops are forwarded from the ex_ctrls flip-flops. The wb_reg also manifested 0% LoV for both simulations. This is because wb_reg is used for WB forwarding, which is only invoked for a store instruction to store the execution into the data cache. Therefore wb_reg has a low vulnerability to radiation hits.

5.3 The Self-Checking Hardened Pipeline Design on OpenRISC Microprocessor

Based on the soft error vulnerability results presented in Figure 5.20, the proposed pipeline protection technique is developed in the OpenRISC 1200 microprocessor [84]. The radiation-hardened processor is then implemented in 65nm technology. The sequential cells, if_insn, id_insn, if_ctrls, id_ctrls, ex_ctrls, RF, $flags$, $op1$, $op2$, and PC are selected for protection since they manifest the highest vulnerabilities. The design of the robust microprocessor pipeline and its operating principles are described in this section.

5.3.1 The Radiation Hardened OpenRISC Pipeline Design

The complete radiation hardened OpenRISC pipeline design is shown in Figure 5.21. The pipeline architecture is broadly divided into a speculative domain and a non-speculative domain. The speculative domain consists of the IF, ID and EX stages of the pipeline. The registers and flip-flops in the speculative domain commit speculative executions, which do not change the architectural state of the pipeline until the execution results are stored in the non-speculative domain in the WB stage. The non-speculative domain consists of the registers updated in the WB stage of the pipeline. These registers, such as the RF and the flag registers, contain the intermediate execution results and architectural states of the pipeline. During the architectural replay execution process, the registers and flip-flops in the speculative domain of the pipeline can be simply flush and do not need to be restored before the re-execution. This is because flushing these registers and retrying the execution will not affect the correct architectural state of the processor. However, the registers in the non-speculative domain of the pipeline stores the state of the processor, therefore they need to be restored to the correct state before any replay executions can be conducted.

The vulnerable registers, if_insn, id_insn, $op1$, $op2$, the RF, and the flag registers are all protected by the SETTOFF-based self-checking radiation register architectures proposed in Chapter 4, Section 4.2.2. The vulnerable flip-flops storing the control signals between each pipeline stage, if_ctrls, id_ctrls, and wb_ctrls, are replaced by the SETTOFF architecture. In order to realise the self-checking capability in these SETTOFF-protected flip-flops, two self-checkers are added, with one shared by the if_ctrls and id_ctrls flip-flops, and the other one shared by the the wb_ctrls flip-flops. The PC register is protected by a TMR architecture since it is the most vulnerable part of the processor.

The architectural replay-based recovery mechanism introduced in Chapter 4, Section 4.2.4.1 is used for recovering the errors detected by the self-checkers. All the vulnerable registers are therefore protected by the basic self-checking architecture illustrated in Figure 4.6 in Chapter 4, apart from the RF and the flag registers. This is because unlike other registers, the RF and the flag registers might not be re-written during the replay operations, thus the erroneous TD-based architecture might not be reset and the errors could remain. Consequently, the RF and flag are protected by the clock gating-based self-checking architecture described in Figure 4.7 in Chapter 4. The clock gating-based architecture allows the $Error\_TD$ signal to reset the corresponding TD architectures in the RF or flags, even when the registers are not updated during the replay operations. It should be noted that since the architectural replay-based recovery mechanism is involved, the glitch filters are not incorporated at the system-level design. As discussed in Chapter 4, Section 4.2.4.1, the glitches in the $Error\_TD$ can only trigger an unnecessary replay operation, but cannot corrupt the correct system executions.

All the $Error\_TD$ signals from self-checkers for the registers and flip-flops in the speculative domain are OR-ed together. The resulting $Error\_TD\_spec$ signal is fed into the pipeline recovery control unit to trigger corresponding replay operations. Similarly, the $Error\_TD$ signals from self-checkers for the registers and flip-flops in the non-speculative domain are also OR-ed together and fed into the control unit. The principle of the recovery operations for the errors detected by the self-checkers is described in Section 5.3.2.3.

The $Error\_SET$ signals generated from the TRD architecture of the SETTOFFs in each register are OR-ed together. Meanwhile, the $Error\_SET$ signals generated from the flip-flops storing the control signals in the IF stage, ID stage, and EX stage are also OR-ed together. The resulting signals from these OR-ed $Error\_SET$ signals are shown by the red dashed arrows. The $Error\_SET$ signals generated by the registers and flip-flops in the speculative domain (including SET_spec1 to SET_spec6) are further OR-ed together, and the final result is fed into a shared error flip-flop for speculative operations. Similarly, the $Error\_SET$ signals generated by the registers and flip-flops in the non-speculative domain (including $SET\_nonspec1$ to $SET\_nonspec3$) are further OR-ed together, and the results are fed into a shared error flip-flop for non-speculative operations. Rather than generating a new delayed clock, we simply used the falling edge of the system clock to drive both of the error flip-flops. As discussed in Chapter 3, Section 3.5, certain

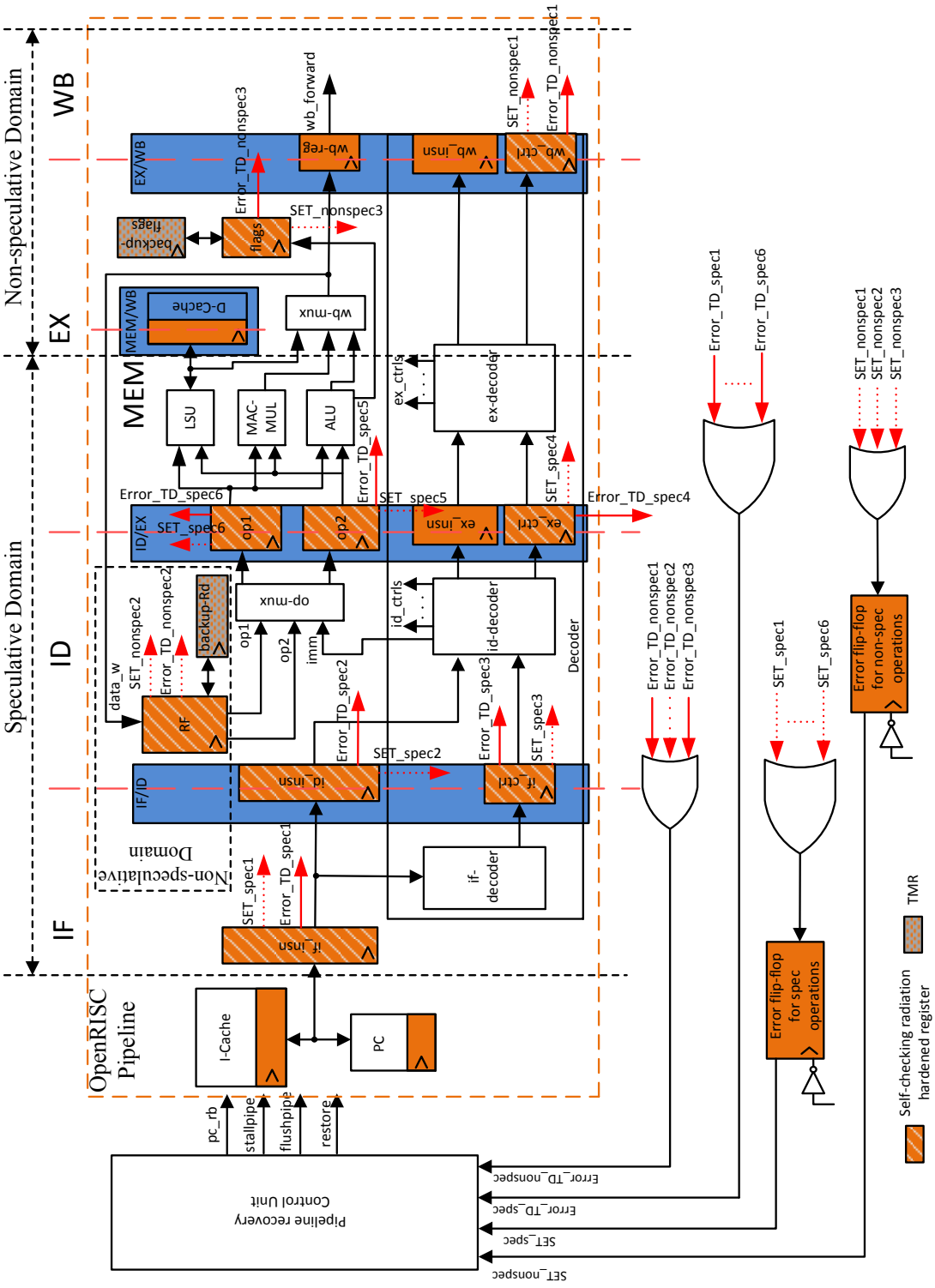

Figure 5.21: The robust pipeline design of the OpenRISC processor

constraints need to apply for the TRD architecture to effectively address SETs. The realisation of a sufficient TRD interval, and the approach for meeting the constraints required by the TRD architecture are further illustrated in Section 5.4.2. Notice that the two error flip-flops only capture SET error signals during the write cycle of the registers when the TRD architectures are enabled. The two error signals, SET_spec and $SET\_nonspec$, are fed into a pipeline recovery control unit which controls the recovery operations within the pipeline. The principle of the recovery operations for the errors detected by the TRD architectures are described in Section 5.3.2.2.

Two back-up registers, $backup\_rd$ and $backup\_flag$, are added to back up the destination register in the RF and the flag register, which are updated by the instructions in the WB stage. The backed-up data is used for restoring the RF and flag registers during the recovery operations for registers that are updated during the WB stage (see Section 5.3.2 for details of the replay operation). Both of the back-up registers are protected by TMR.

5.3.2 Operating Principles of Error-Tolerance in the Radiation-Hardened Pipeline Architecture

The location of the radiation-induced transient faults inside the pipeline could be random. However, they can be categorized into 3 types as follows. The recovery mechanisms for the errors in each type are described separately.

1. SEUs that corrupt the data stored in the registers.
2. SET pulses that are produced by the combinational logic gates.
3. Transient faults occurring in the redundancy circuitry added for error-tolerance.

5.3.2.1 Recovery Mechanism for Type (1) Errors

As discussed Chapter 3, Section 3.3.1 and Section 3.4.1, the errors of type (1) can be detected and recovered on the fly by the TD-based architecture within the SETTOFF. Therefore, no extra recovery operations are required for these errors.

5.3.2.2 Recovery Mechanism for Type (2) Errors

For type (2), the SETs occurring at each stage of the pipeline will be detected by the radiation hardened registers incorporated in the corresponding stages, if they are captured. The architectural recovery operation is then required for correcting such SETs. The PC moves forward and is check-pointed as it passes along each pipeline stage. This allows the instruction at any stage of the pipeline to be replayed. As shown

in Figure 5.21, two SET error signals, *SET_spec* and *SET_nonspec*, are generated for the SETs captured by the speculative registers and non-speculative registers, respectively. The two error signals trigger two types of architectural replay recovery operations which re-execute the instruction in either the EX stage or the WB stage, depending on the following three cases:

Case (1): If an SET occurs in the IF or ID stage of the pipeline, it will be detected by the SETTOFF-based sequential cells incorporated in the speculative domain of the pipeline. The detection of such SETs will assert the *SET_spec* error signal, which will trigger a replay operation at the beginning of the following cycle, before the SET contaminates the register file and the flag registers in the non-speculative domain. Since in this case, the SET only corrupted the speculative operations, and none of registers storing the architectural state of the pipeline are affected, the replay operation for case (1) will simply flush the entire pipeline, re-fetch and re-execute the instruction in the execution stage (i.e. the instruction in *ex_insn*.) to overwrite the capture SET.

Figure 5.22 shows an example timing diagram for the recovery process of an SET captured by the *id_insn* (Case (1)). *PCn* stands for the PC for the instruction *INSNn*. The *INSN3* captured by the *id_insn* at the rising clock edge of cycle 3 is corrupted by an SET occurring in the decoder in the IF stage. If the pipeline is unprotected, the ID stage of the pipeline will decode the erroneous *INSN3* stored in the *id_insn* during cycle 3, and generate erroneous control signals to corrupt the execution stage. The corrupted execution results will then be forwarded to the registers in the non-speculative domain (RF, flag registers, and *wb_reg*). However, in the protected pipeline, the captured SET is detected by the TRD architecture in *id_insn* at the falling clock edge of cycle 3, which asserts the *SET_spec* signal in the error flip-flop for speculative operations. The *SET_spec* signal then triggers the recovery operation for Case (1). The recovery operation asserts the *flushpipe* signal, which flushes all the pipeline registers at the rising clock edge of cycle 4 to prevent the erroneous execution from propagating. The *stallpipe* signal is also asserted to stall the pipeline and disables all the registers from updating the results from executing the instruction *INSN3* stored in *ex_insn* in cycle 4. The PC is rolled back to the PC in EX stage (i.e. the *PC2*), by the *PC_rb* signal. When the new PC value becomes available at cycle 6, the pipeline starts fetching the new instruction and normal operation resumes (*NOP* stands for the NOP instruction that does not contain any valid executions). The SET is not repeated in the IF stage during the re-execution in cycle 8, so that *id_insn* captures the correct *INSN3* after the replay. The recovery process for Case (1) ensures that the errors in all pipeline registers in the speculative domain are overwritten during the re-execution to prevent them from corrupting the WB stage.

Case (2): If an SET occurs in the EX stage, it can corrupt the non-speculative registers, such as the RF, the flag register, and the *wb_ctrl* flip-flops during the WB cycle. The recovery operation in this case will re-execute the instruction in the WB stage (i.e.

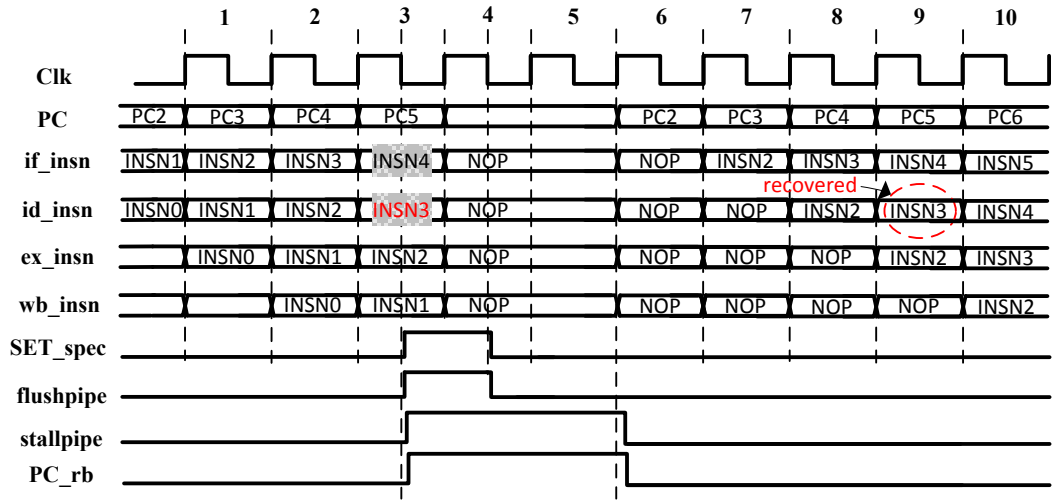

Figure 5.22: Timing diagram of the pipeline recovery operation for Case (1)

the instruction in *wb_insn*) to re-write the RF, the flag register, and the *wb_ctrl* flip-flops. However, unlike the replay operation in Case (1), the intermediate data and the architectural state of the pipeline might have been corrupted by the SET. These data and pipeline states may be used as operands or other operation conditions during the re-execution. Therefore for Case (2), the RF and the flag register need be restored using their back-up registers before the recovery operation is committed.

Figure 5.23 depicts an example timing diagram for the recovery process of an SET captured by the RF (Case (2)) [2]. When the execution in the EX stage is about to write its results into the destination register (*Rd*) in RF and the flag register, both *Rd* and the flag register are backed-up into their back-up registers before they are updated. *resultn* and *flagn* stand for the execution results and the flags generated by the *INSNn* in the EX stage. In cycle3, the captured *result4* in *Rd* is corrupted by an SET occurring in the EX stage. The *SET_nonspec* signal is asserted upon detection which invokes the recovery process for Case (2). The pipeline is flushed and stalled while the PC is rolled back to the PC value in the WB stage (i.e. *PC4*), to re-fetch and re-execute the instruction in *wb_insn* (i.e. *INSN4*) which corrupts the RF. In addition, *Rd* and the flag register are restored by the back-up registers in cycle 4 to ensure that the system state is consistent as the original execution during the re-execution. In cycle 10, *Rd* is re-updated and recovered by the re-execution. The recovery process for Case (2) ensures that the all SETs captured in the WB cycle can be mitigated.

Case (3): A third circumstance can happen if a captured SET is detected when a branch instruction is executing. Figure 5.24 illustrates the timing diagram for Case (3). *PC2(brc)* stands for the PC for the branch instruction *brc*. *PC100* is the branching address. In OpenRISC, the branch operation is committed in the ID stage to force the

[2] Control signals (*flushpipe*, *stallpipe*, *PC_rb*) are not shown in Figure 5.23 since they operate the same as shown in Figure 5.22.

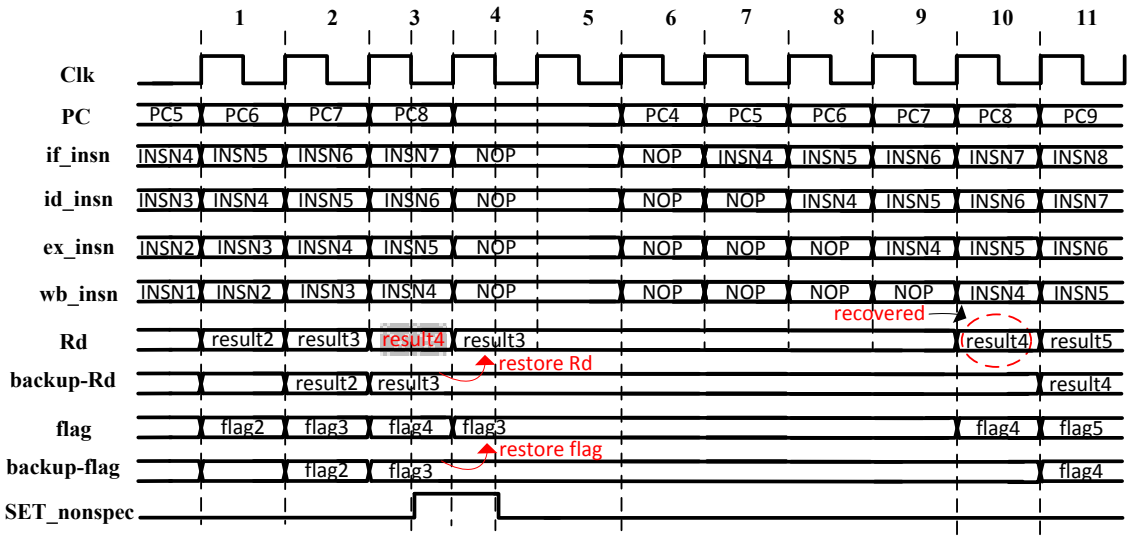

Figure 5.23: Timing diagram of the pipeline recovery operation for Case (2)

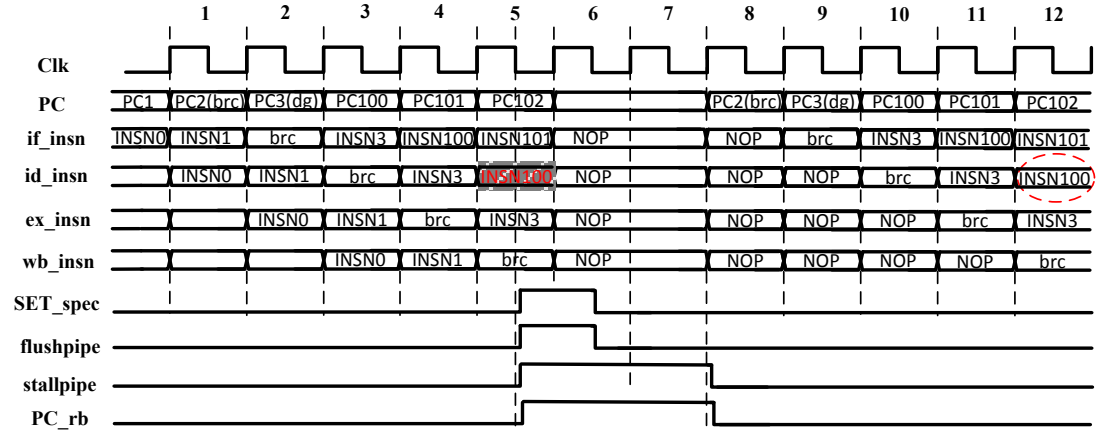

Figure 5.24: Timing diagram for pipeline register recovery with branch instruction (Case 3)

PC to jump to the branching address. Therefore a dangerous PC (denoted by *PC3(dg)*) is forwarded into the pipeline before the branching PC, to fetch the *INSN3*. A dangerous situation can happen if an error is detected when *INSN3* is in the *ex_insn*. In this case, re-fetching the instruction in *ex_insn* only forwards *PC3(dg)* into the pipeline, therefore the branch instruction stored in *PC2(brc)* will not be executed during the replay. This results in the branch operation not being detected during the re-execution, such that the processor fails to jump to the branching address. The problem is solved by monitoring the branch operation within the pipeline, and when such situation occurs, the branch PC in the WB stage (*PC2(brc)* in this case) are replayed as is shown in Figure 5.24. The error is corrected in cycle 11 in *id_insn*.

Most of the errors detected by the TRD architecture are transient faults which may not occur again during the re-execution. In such cases, the recovery process can successfully

recover the error at one time. If other SETs occur during the replay operation, they will be detected and will trigger further replays until no errors are detected. However, the recovery mechanism is not feasible for timing errors. This is because timing errors may repeat and be detected again during the recovery process to trigger another recovery operation. To solve this problem, the number of the repeat recovering processes can be recorded such that when it exceeds a certain value n, which is programmable, the error is considered a timing error. Certain frequency or voltage tuning techniques may be required for recovering from timing errors during re-execution. The usage of the potential timing error-tolerance capability will be addressed as future work, discussed in Chapter 6, Section 6.2.1.

5.3.2.3 Recovery Mechanism for Type (3) Errors

The errors of type (3) include errors that corrupt the TRD architecture, errors that corrupt the TD-based architecture, and errors that corrupt the self-checker.

The errors that corrupt the TD-based architecture are detected by the self-checkers, and are recovered by the replay-based recovery mechanism introduced in Chapter 4, Section 4.2.4.1. Similar to the replay recovery mechanism for the errors of type (2), the errors generated from the speculative domain will assert the $Error\_TD\_spec$ signal (see Figure 5.21), which will then trigger a replay execution to flush the pipeline, re-fetch and re-execute the instruction in the EX stage. The errors from the non-speculative domain will assert the $Error\_TD\_nonspec$ signal, which triggers a replay execution that replay the instruction in the WB stage. During both replay operations, all the pipeline registers and flip-flops storing the control signals will be re-written, such that all the TD-based architectures in these cells are reset. The errors generated from the TD-based architecture of these cells will be recovered after the replay, when normal operation can resume. Nevertheless, if an error is detected from the TD-based architectures in the RF or the flag register, the $Error\_TD$ signal will also trigger a reset signal to reset the TD-based architectures in the corresponding register which generates the error, besides triggering the replay operation. This is because the RF and the flags might not be updated during the replay, such that the reset of TD may not be triggered automatically. The erroneous TD-based architecture can stay corrupted unless the reset operation for the TD is explicitly executed.

The errors corrupting the TRD architecture and the self-checker may produce a faulty error signal triggering an unnecessary recover process. However, such errors do not corrupt the pipeline execution results. Therefore, no extra recovery operations are involved.

5.4 Experimental Methodology and Implementation Process

The OpenRISC processor with the proposed radiation hardened pipeline architecture was implemented in STMicroelectronics 65nm technology [81] [3]. The transistor-level implementation of SETTOFF and the self-checker, which construct the radiation hardened register architecture, have been characterized for the 65nm technology as new cells using Synopsys Liberty NCX [90]. The behavioral model of the OpenRISC 1200 processor was re-designed to incorporate the new pipeline architecture described in Section 5.3, which was then synthesized to gate-level descriptions using the characterized 65nm technology library. The 32-bit pipeline registers and the registers in RF were replaced by the 32-bit radiation hardened registers. Gate-level simulation was carried out, based on the ORPSoc platform which provides the smallest-possible reference system for testing the processor [89]. This section describes the details of the experimental methods for implementing the processor.

5.4.1 Cell Characterisation using Synopsys Liberty NCX

The two versions of the SETTOFF architecture introduced in Chapter 3 are custom transistor-level designs. In order to incorporate SETTOFFs into the OpenRISC microprocessor and carry out system-level evaluation through gate-level simulations, they need to be characterised as new cells to be added into a technology library. Synopsys Liberty NCX, [90], is a software tool that can generate a library in Liberty (.lib) format by extracting the essential timing and power information through a number of SPICE simulations of the cells. The transistor-level design of the SETTOFFs are translated into Liberty format and added into the STMicroelectronics 65nm standard cell library by using Liberty NCX.

5.4.1.1 Cell Characterisation Flow

Figure 5.25 shows the data that is required for cell characterisation and the generation of the new library. For translating a transistor-level design of a cell into the Liberty format, the SPICE netlist of the cell, and the transistor model file that contains the process information of any elements constructing the cell are compulsory. The SPICE netlist of the cell needs to be contained within a sub-circuit. The library template, and the cell templates for each cells to be characterised are also prerequisite. The library template file specifies the library characterisation parameters, such as units, delay threshold, and slew thresholds. An example of the library template is provided in Figure 5.26. The delay threshold specifies the points that the propagation delays are measured (40% of the

[3] BUt is not actually fabricated

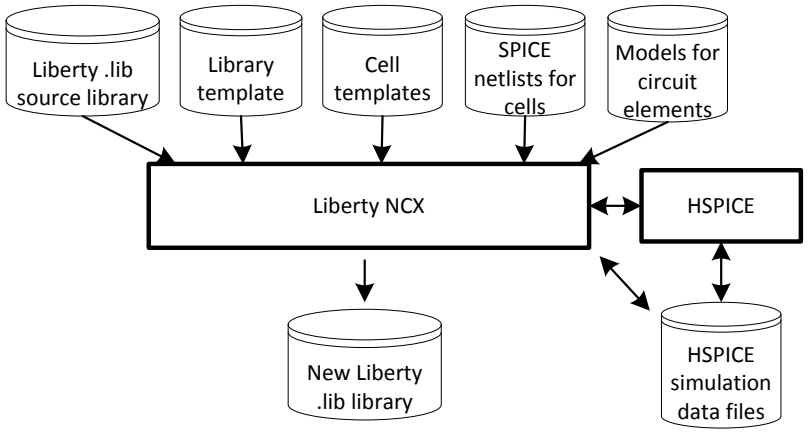

Figure 5.25: The data required for NCX characterisation [90].

signal voltage for rising transitions, and 60% of the signal voltage for falling transitions). The slew threshold specifies the transition times to be measured, which is from 20% to 80% of the signal voltage. The list in the *do* statement specifies all cells that need to be characterised by NCX.

```
*Unit defination
nom_voltage : 1.2000000 ;
nom_temperature : 25.0000000 ;
time_unit : 1ns ;
voltage_unit : 1V ;
current_unit : 1mA ;
pulling_resistance_unit : 1kohm ;
capacitive_load_unit : 1.0000000 pf ;

*delay threshold
input_threshold_pct_fall : 60.0000000 ;
input_threshold_pct_rise : 40.0000000 ;
output_threshold_pct_fall : 60.0000000 ;
output_threshold_pct_rise : 40.0000000 ;

*slew threshold
slew_lower_threshold_pct_fall : 20.0000000 ;
slew_lower_threshold_pct_rise : 20.0000000 ;
slew_upper_threshold_pct_fall : 80.0000000 ;
slew_upper_threshold_pct_rise : 80.0000000 ;

*List of cells to be characterised
do {
  inv1
  inv2
}
```

Figure 5.26: Example of a library template.

The cell templates specify the cell-level parameters, such as the cell area, the cell function descriptions in Liberty format, the input and output pins of the cell. Figure 5.27 shows an example of the cell template. The function of the cell (inverter) is described using a simple Boolean logic function. For more complicated combinational cells, the

functions may need to be described in truth tables. For sequential cells, *statetable*, ff, or *latch* groups are used for describing the functions. The details for the Liberty format description of the functions of complicated cells will be introduced later in Section 5.4.1.2 and Section 5.4.1.3.

```
area : 3.6400000 ;
cell_footprint: inv ;
pin A {
    direction : input ;
    }
pin Y {
    direction : output ;
    function : !A ;
}
```

Figure 5.27: Example of a cell template.

The source library is optional, but can be used to generate the library template and the cell templates if provided. The designer can then specify whether to use the templates generated from the source library, or self-defined templates during the characterisation process. When creating a new library, self-defined templates must be provided to specify the parameters at both library-level and cell-level. When adding new cell models to a library or modifying the existing cell models in the library, the templates generated from the source library are normally used to maintain the coherence of the library attributes between the new output library and the source library.

During the characterisation process, NCX sensitises all the cell-arcs that are required to produce simulation measurements of the desired characteristics, such as the delay and slew rates of each cells. The sensitisations of the cell-arcs are generated from the cell function, or a sensitisation explicitly specified in the cell template file. The SPICE netlists for simulating all the cell-arcs are then created and sent to the HSPICE simulator. HSPICE simulates all the cell-arcs based on modelling the actual transistor-level silicon behavior specified in the transistor model file. After all the simulations are performed, the cell models are extracted and translated into the Liberty format. The new library that contains the new characterised cells is finally written out. Liberty NCX can describe a variety of models in the output library. Composite Current Source (CCS) format, for instance, is a newer model which can be used for timing, power, and noise analysis for 90nm technology or below. NCX can also acquire the models in Nonlinear Delay Model (NLDM) format, which is a older format based on voltage sources. The output library in Liberty (.lib) format can be translated into database format (.db), or HDL cell description models (.vhd or .verilog) using the Synopsys library compiler. The database format can be used by the Synopsys Design Compiler for RTL synthesis. The HDL cell description models can be used in gate-level simulations after the synthesis, to verify the system-level functionality.

5.4.1.2 Characterisation of the Complete SETTOFF

The complete SETTOFF1 and SETTOFF2 architectures are characterised as new cells and are added into the STmicroelectronics 65nm library. The SPICE models of both SETTOFFs, and device models from the the 65nm technology process are used for the characterisation. The library template used is generated from the 65nm source library. The *do* lists in the library template (refer to Figure 5.26) are modified to include the SETTOFF1 and SETTOFF2 for characterisation. The cell templates used for characterising the SETTOFFs are also generated from the source library, and are consistent with the cell templates of the main flip-flops that constructed the SETTOFFs. This is because both SETTOFFs perform exactly the same as the conventional main flip-flop during normal operation. The templates for the conventional flip-flops can correctly describe the function of the SETTOFFs in normal operation, and therefore proper sensitisations of the cell-arcs can be determined for characterising the SETTOFFs. Only the cell area specified in the template file is manually modified according to the ratio of the transistor count between the main flip-flop and the SETTOFFs. The characterisation generated both the CCS and NLDM timing and power behavioral models. The output library with the new SETTOFF cells can be used for synthesising the microprocessor to incorporate the SETTOFFs into the pipeline. The gate-level description of the microprocessor can be used for carrying out accurate timing and power analysis at the system level.

5.4.1.3 Characterisation of the Error-Tolerance Circuitry in SETTOFF

The characterisation of the complete SETTOFFs captures the accurate timing and power information of the transistor-level design during normal operation (error-free), and preserves this information in the gate-level models for gate-level simulation. However, the error-tolerance functionality and capability of the SETTOFFs were not captured. This is because the characterisation used the template file of the main flip-flop, in which only the function of a conventional flip-flop is specified. In order to characterise the error-tolerance functionality of the SETTOFFs, a full description of the cell functions in Liberty format for both error-free conditions and error conditions would be required. However, the SETTOFF design is a complex cell, whose functionality cannot be fully specified in Liberty format. To address this problem, the error-tolerance circuitry that constructed the SETTOFFs are separated from the whole circuitry, and are treated as individual cells for characterisation. This allows the functions of the error-tolerance circuitry to be described separately from the main flip-flops. The details of characterising the error-tolerance circuitry of both SETTOFF1 and SETTOFF2 are explained as follows:

The error-tolerance circuitry in SETTOFF1 which consists of the DC generator, the TD, and the correction XOR-gate are separated and are constructed as a new cell named

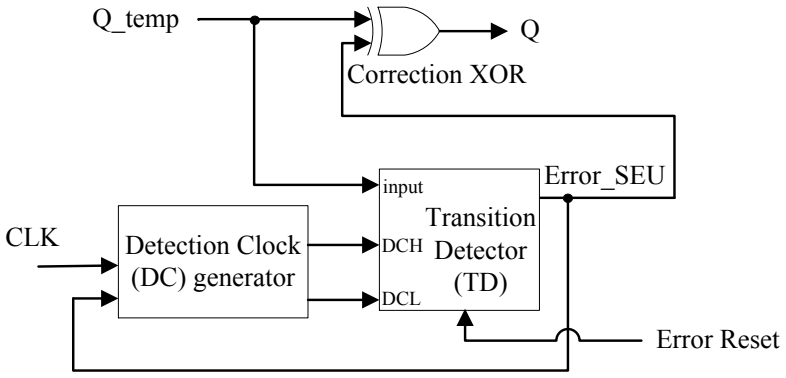

Figure 5.28: The SPICE netlist of the TDXOR1 in SETTOFF1.

TDXOR1. The SPICE netlist of the TDXOR1 shown in Figure 5.28 is imported into NCX for characterisation. The source library, the library template, and the device mode file used for characterisation are all the same as the ones used for the characterisation for the complete SETTOFF1. However, the cell template is self-defined to include the function of the TDXOR1 specified in Liberty format, which is shown in Figure 5.29. Since the TD is a retention cell (not purely combinational), its function needs to be described using a state table. The definition of the values in the state table can be found in Table 5.2. *Qtemp*, *CLK*, and *Error_reset* are the input pins of the cell, and *IError_SEU* is the internal node of the TDXOR1. The values in the first 3 columns of the state table in Figure 5.29 specify all the possible input combinations of the 3 input pins. The last 2 columns leading by the colons represent the current state and the next state of the node *IError_SEU*, respectively. The current and next states shown in node *IError_SEU* match the correspondence input values. The function of the output pin *Q* is specified by a Boolean expression of the inputs and the internal node.

Table 5.2: State table definition in Liberty format.

Value	H	L	-	R	F	N	H/L	L/H
Meaning	high	low	Don't care	rising edge	falling edge	hold	expands to both L and H	expands to both H and L

Because NCX does not support modelling complex sequential cells, the TDXOR1 will be modelled as a conventional sequential cell (flip-flops or latches) during the characterisation. The sensitisations of the cell-arcs are then performed based on the conventions of flip-flops or latches, by regarding the *Qtemp* as a clock pin. Such sensitisations will produce certain meaningless cell-arcs such as the setup time and the Clock-to-Q delay. These cell-arcs cannot be measured for the TDXOR1, so they need to be removed

```
statetable ( Qtemp CLK Error_reset , IError_SEU ) {

    table :     -    -      L          :  -  :  L , \
                -    H      H          :  L  :  N , \
                R    L      H          :  L  :  H , \
                F    L      H          :  L  :  H , \
                H    L      H          :  L  :  N , \
                L    L      H          :  L  :  N , \
                R    -      H          :  H  :  L , \
                F    -      H          :  H  :  L , \
                H    -      H          :  H  :  N , \
                L    -      H          :  H  :  N  ;
   }

pin Qtemp {
  direction : input ;

}

pin CLK {
  direction : input ;
}

pin Error_reset {
  direction : input ;
}

pin Error_SEU {
  direction : internal ;
  internal_node : IError_SEU ;
}

pin Q {
  direction : output ;
  state_function : !Qtemp*Error_SEU+Qtemp*!Error_SEU ;
}
```

Figure 5.29: Function description of the TDXOR1 in Liberty format.

manually before the SPICE netlists for simulating these arcs are created and sent to the HSPICE simulator. After removing the meaningless cell-arcs, NCX will simulate all the meaningful cell-arcs of TDXOR1, and extract the required information in Liberty format. The output library will consist of the TDXOR1 as a new cell model.

Similarly, the error-tolerance circuitry in SETTOFF2 which consists of the optimised TD and the correction XOR-gate is separated from the entire cell. The new cell shown in Figure 5.30 is named as TDXOR2. Figure 5.31 shows Liberty-format function description of the TDXOR2 in the cell template for characterising the TDXOR2. The characterisation process for the TDXOR2 is the same as that for characterising the TDXOR1. Because the full functions of the cell are captured during the characterisation, and are preserved in the new cell models in the output library, the Liberty format of the cell models of TDXOR1 and TDXOR2 can be translated into the HDL description models for gate-level simulation. For realising the gate-level simulation of both SETTOFFs, the HDL descriptions of the TDXOR1, TDXOR2, and the main flip-flops are manually connected to construct the SETTOFFs. The full functionalities under both error-free conditions and error conditions for both SETTOFFs can then be realised.

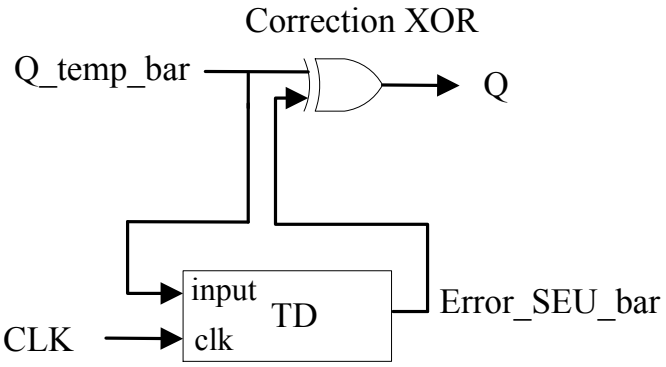

Figure 5.30: The SPICE netlist of the TDXOR2 in SETTOFF2.

```
statetable  Qtemp_bar CLK   IError_SEU_bar  {

  table :       -      H     : - : L , \
                R      L/H   : - : H/L , \
                F      L/H   : - : H/L , \
                H      L     : - : N , \
                L      L     : - : N  ;
}

pin Qtemp_bar {
  direction : input ;
}

pin CLK {
  direction : input ;
}

pin Error_SEU_bar {
  direction : internal ;
  internal_node : IError_SEU_bar ;
}

pin Q {
  direction : output ;
  state_function : !Qtemp_bar*Error_SEU_bar+Qtemp_bar*!Error_SEU_bar ;
}
```

Figure 5.31: Function description of the TDXOR2 in Liberty format.

5.4.1.4 Characterisation of the Self-Checker

The same method for characterising the TD-based circuitry described in Section 5.4.1.3 is also used for characterising the self-checker. As shown in Figure 4.3 in Chapter 4, the self-checker consists of a TD-checker, an XOR-gate, and a glitch filter. Since the glitch filter is not incorporated into the pipeline of the radiation-hardened processor (See Section 5.3.1), only the TD-checker and the XOR-gate are combined and are characterised as one new cell in the 65nm technology library. The Liberty-format function description of the combined TD-checker and the XOR-gate is shown in Figure 5.32. Refer to

the circuit schematic of the TD-checker shown in Figure 4.4, Chapter 4, Pin Q is the input of the TD-checker connected to the output of the SETTOFF. *rising_tran* and *falling_tran* are the two outputs of the TD-checker. *Error_TD* is the output pin of the self-checker generated from the XOR-gate. Again, since the full functions of the cell are captured during the characterisation, the Liberty format of the cell model of the self-checker can be translated into HDL models for gate-level simulation.

```
statetable  Q   CLK   IFalling_tran   IRising_tran  {

  table :   -    H      : - : L       : - : L, \
            R    L/H    : - : L       : - : H/L, \
            F    L/H    : - : H/L     : - : L, \
            H    L      : - : N       : - : N, \
            L    L      : - : N       : - : N  ;
}

pin Q {
  direction : input ;
}

pin CLK {
  direction : input ;
}

pin falling_tran {
  direction : internal ;
  internal_node : IFalling_tran ;
}

pin rising_tran {
  direction : internal ;
  internal_node : IRising_tran ;
}

pin Error_TD {
  direction : output ;
  state_function : !falling_tran*rising_tran+falling_tran*!rising_tran ;
}
```

Figure 5.32: Function description of the self-checker in Liberty format.

For incorporating the characterised self-checking architecture into the register shown in Figure 4.6 in Chapter 4, the parity checker is constructed by a XOR-tree using structural HDL descriptions. The HDL description of the self-checker is then manually placed into the gate-level description of the SETTOFF-protected register to construct the whole self-checking register architecture. The self-checking register architecture can then be incorporated into the gate-level description of the processor pipeline to carry out gate-level simulations for the whole OpenRISC processor.

5.4.2 Setting TRD and TD Intervals and Clock Management

In the radiation-hardened OpenRISC processor implementation, only one clock is used for driving both the system flip-flops (by using the positive clock edge), and the error flip-flops in the TRD architecture (by using the negative edge of the clock). The TRD and TD intervals of the incorporated SETTOFF architectures are hence decided by the

frequency and the duty cycle of the clock. In order to be consistent with the circuit-level and register-level evaluations presented in Chapter 3, and Chapter 4, a 185MHz clock is used for driving the processor.

The choice of the TRD and TD intervals is based on the purpose of providing desirable SET and SEU coverage. The two intervals formed by the two clock phases represent a trade-off between SET- and SEU-tolerant capabilities. A bigger TRD interval allows more SETs with bigger widths to be addressed, while a bigger TD interval can detect and correct more SEUs on the fly. Since the SEUs that are not corrected during the TD interval will still be addressed during the TRD interval, the aim is then to provide a sufficient TRD interval to efficiently reduce both the SET and SEU failure rate of the SETTOFF. The widths of the potential SET pulses are the main factors for determining the TRD interval. As reported in [79], the mean and the standard deviation of the width of the SET pulses in 65nm technology are $530ps$ and $150ps$, respectively. According to the circuit-level evaluation results in Chapter 3, Section 3.7.5.2, a minimum $800ps$ TRD interval is chosen because it can reduce the SET failure rate of the SETTOFF to 0 in 65nm technology. With a $800ps$ TRD interval, the SEU failure rate of the SETTOFF is also reduced to 0 since it covers all the correction glitches.

In order to achieve a minimum of $800ps$ TRD interval with a 185MHz clock, the circuit-level TRD constraints described in Chapter 3, Section 3.5, and the register-level TRD constraints described in Chapter 4, Section 4.2.3 need to be satisfied. The TRD interval is formed by the delay element δ, which equals the high phase of the clock since the negative edge of the system clock is used for driving the TRD architectures. The maximum delay of the comparator (including the XOR comparator and the OR-tree) of the TRD architecture at register-level is $400ps$, which is obtained from Equation 4.1 in Chapter 4, Section 4.2.3. A $1.5ns$ delay element δ is chosen, which can satisfy the minimum TRD interval requirement and other TRD constraints described in Chapter 3, Section 3.5. The duty cycle, $Duty_{clk}$, of the 185MHz system clock is therefore shown by Equation 5.2.

$$Duty_{clk} = \frac{D_{hclk}}{P_{clk}} = \frac{1.5ns}{5.4ns} = 27.8\% \tag{5.2}$$

where D_{hclk} is the high phase of the clock, and P_{clk} is the clock period.

Finally, buffers are inserted into the shortest path of the combinational logic in each pipeline stages to satisfy the shortest path constraint of the TRD technique for all the incorporated radiation-hardened cells. The buffers that provide the maximum delay with minimum power consumption and area occupation are chosen to minimise the incurred overhead.

5.5 Evaluation Results and Comparative Analysis

The reliability and the implementation overheads of the proposed radiation-hardened processors are evaluated through gate-level simulations. The results for the SETTOFF2-based robust processor architecture are explicitly presented, and compared with previous pipeline protection techniques.

5.5.1 Reliability Evaluation for the Radiation-Hardened Processor

The transient fault injection technique and the simulation methodology used for analysing the error vulnerability of the original OpenRISC described in Section 5.2 are used for evaluating the reliability of the radiation-hardened processor. However, besides injecting transient faults into the original cells using the methods described in Section 5.2.2, the same method is also used to inject transient faults into the redundant circuitry of the processor added for error-tolerance. To be specific, the transient faults injected into the redundant circuitry include:

- The SEUs injected into the transition detectors to corrupt the TD-based architectures in SETTOFFs.
- The SEUs injected into the error flip-flops to corrupt the TRD-based architectures in SETTOFFs.
- The SEUs injected into the TD-checkers to corrupt the self-checkers in the self-checking registers.
- The SEUs injected into the backup registers for the RF and flag register.

Three programs, the quicksort and tak programs used in the vulnerability analysis in Section 5.2.3, and a matrix multiplication program, are used for the fault simulation. The evaluation results are compared with the reliability results of an unprotected OpenRISC processor. Table 5.3 shows the number of transient faults injected into each processor and the visible errors occurring in the data memories after running the programs. A total of 4050 identical faults (including both SETs and SEUs) are injected into the original cells of each of the two processors. These faults incur 306, 305, and 354 errors in the original process for the quicksort, tak, and matrix multiplication programs, respectively. All these errors however, are mitigated in the proposed OpenRISC processor. In addition, the transient faults injected into the redundancy circuitry in the proposed processor are also tolerated and did not induce any soft errors in the outputs of the programs.

Table 5.3: Reliability analysis results for the radiation hardened OpenRISC

Program	Faults injected into the original cells		Faults injected into the redundancies
	4050		1411
	Errors in original OpenRISC	Errors in radiation hardened OpenRISC	Errors in radiation hardened OpenRISC
quicksort	306	0	0
tak	305	0	0
matrix multiplication	354	0	0

5.5.2 Implementation Details and Error-tolerance Overheads

The implementation details of the SETTOFF2-based radiation-hardened OpenRISC processor are summarised in Table 5.4. The results come from evaluating only the core of the processor; the power consumption and area of other parts of the processor, such as the caches and memory is not considered. The processor is implemented in 65nm technology with 1.2V supply voltage and 185MHz clock frequency. The total error-tolerance area overhead and power overhead for the radiation-hardened OpenRISC processor are 26.7% and 17.8%, respectively, and break down as follows: the incorporated replay recovery architecture and the TRD circuitry of all SETTOFF2s incur 13% more area and 2.4% more power consumption. A total of 956 buffers are inserted into the pipeline to achieve the timing constraint for the TRD architectures. The inserted buffers incur 1.5% extra area and 0.5% more power consumption in respect to the whole processor core. A total number of 1219 SETTOFF2s and 39 self-checkers are incorporated. These self-checking architectures incur an area overhead of 12.1% and a power consumption overhead of 14.7%.

5.5.3 Comparative Analysis with Razor and SEM/STEM Pipeline Protection Techniques

Both the RazorII pipeline protection technique and the proposed pipeline protection technique utilized an architectural replay recovery operation, which consumes extra operation cycles and causes Instruction Per Cycle (IPC) overheads. However, in the proposed pipeline protection technique, most of the SEUs are detected in the TD interval and are recovered on-the-fly. The replay operation is only triggered for correcting the captured SETs or the corrupted TD architectures during the TRD interval. RazorII uses a conventional replay recovery mechanism and all the errors are recovered through re-execution. Therefore, the proposed pipeline protection incurs a much smaller IPC overhead than the conventional replay recovery mechanism used in RazorII. In addition,

Table 5.4: Radiation hardened processor implementation details

Technology node	65nm
Frequency	185MHz
Supply Voltage	1.2V
Temperature	25C
Total Area (μm)	121939
Total Power (mW)	22.4
No. of SETTOFF2s incorporated	1219
No. of self-checkers incorporated	39
Area Overhead of the self-checking arch.	12.1%
Power Overhead of the self-checking arch.	14.7%
Replay Recovery Area Overhead	13%
Replay Recovery Power Overhead	2.4%
Number of Buffers inserted for TRD	956
Buffer Area Overhead	1.5%
Buffer Power Overhead	0.5%
Total Error-tolerance Area Overhead	26.7%
Total Error-tolerance Power Overhead	17.8%

the system-level area and power consumption overheads for the proposed protection technique to protect the pipeline registers (except the RF and flags) are comparable with the RazorII technique. The SEM/STEM pipeline protection technique incurs much bigger overheads that are similar to TMR.

On the other hand, RazorII and SEM/STEM pipeline protection techniques cannot protect the WB stage of the pipeline and the registers that store the architectural states of the system during the WB stage (e.g. the RF and the flag registers.). This is because RazorII cannot efficiently address SEUs, and the error-tolerance overheads are too big for SEM/STEM to protect the RF. RazorII and SEM/STEM techniques use a stabilizer buffer or register before the WB stage to make the WB stage not timing-critical. Therefore, the TEs occurring in the WB stage are not considered. The stabilizer buffer or the register also guarantee that no errors are forwarded into the RF during the recovery process since the RF does not have SET/TE-tolerant capabilities (see [42] [41] [44]). However, SETs occurring during the write cycle of WB stage can still corrupt the registers that are updated during the cycle. The proposed pipeline protection eliminates this extra stabilizer and shrinks the pipeline depth compared with the previous pipeline protection techniques. Both the SETs occurring during the write cycle of WB stage, and the SEUs occurring in the registers updated during the write cycle are tolerated by the incorporated robust registers.

5.5.4 Comparative Analysis with ECC-based RF protection technique

It has been demonstrated in Chapter 4, Section 4.4 that the proposed radiation hardened register can efficiently protect the RF. Compared with the traditional ECC-based RF protection technique, the proposed RF protection significantly improved the MBU-tolerance capability since each bit inside the protected RF has its own built-in error-tolerance circuitry. In addition, previous research indicates that the majority of MBUs occurring in the RF are caused by the captured SETs originated in the combinational gates (such as the read and write logic) in the RF [56]. This is because that the combinational logic has a high degree of fanout, and the majority of the cell area within the RF is consumed by the read and write logic. As a result, the proposed pipeline technique can provide a noticeably better error-tolerance than ECC, since ECC does not have SET-tolerant capabilities to combat these SET-induced MBUs.

As reported in Chapter 4, Section 4.4.5, the proposed technique requires much less power consumption and delay overhead for protecting the RF, compared to the ECC technique, but it occupies more chip area.

5.6 Concluding Remarks

This chapter proposed a radiation hardened pipeline architecture based on the self-checking SETTOFF techniques proposed in Chapter 3 and Chapter 4. In order to achieve a cost-effective pipeline protection, a gate-level transient fault injection technique is developed and used for analysing the soft error vulnerability of the OpenRISC processor. Based on the analysis results, the most vulnerable registers and flip-flops within the pipeline are selected to protect using the SETTOFF-based self-checking architectures. The entire pipeline, including the pipeline registers and the RF are protected. The errors occurring in each stage of the pipeline are addressed by the radiation hardened cells incorporated in the corresponding stage. The SEUs occurring within the protected sequential cells will be detected and corrected on the fly at the circuit level. Other harmful transient faults within the pipeline, such as the captured SETs, and the transient faults corrupting the redundant circuitry will be recovered by an architectural replay recovery mechanism.

The proposed robust pipeline architecture is implemented in a 32-bit OpenRISC microprocessor in 65nm for evaluation. The evaluation results show that it can effectively mitigate both SEUs and SETs occurring in different parts of the pipeline. The reliability of the proposed pipeline protection is compared with previous techniques. The comparatively analysis shows that the proposed technique can provide noticeably higher level of reliability for different parts of the pipeline, with less or comparable overheads. It can

overcome the drawback of most previous pipeline protection techniques and achieves a complete and cost-effective protection.

Chapter 6

Conclusions and Future Work

With technology scaling, reliability issues caused by radiation-induced soft errors are becoming increasingly severe. Sea-level non-safety-critical applications are facing a serious challenge, not only because they are becoming more vulnerable to particle strikes, but also because the existing error-mitigation techniques cannot provide effective and affordable protection, especially for general logic. The contributions in this thesis provide novel and cost-effective techniques to achieve soft error mitigation in general logic. The techniques address significant limitations of most previous techniques, and are able to provide complete protection for microprocessor pipelines. Meanwhile, the proposed techniques require lower or comparable overheads compared with previous techniques. The contributions of the thesis are listed in the next section followed by some proposed future research directions.

6.1 Conclusions and Contributions

All the objectives of this research have been clearly listed in Chapter 1, Section 1.6:

1. *Develop cost-effective soft error-mitigation techniques suitable for protecting the general logic of non-safety-critical applications. Try to overcome the significant limitations of the previous techniques by providing acceptable trade-offs between the reliability and error-tolerance overheads.*

2. *Realise self-checking capabilities in the developed techniques. Allowing the techniques to be robust to both the errors occurring in the original circuitry and the errors occurring in the redundant parts. Minimise the overheads required by the self-checking feature to keep the technique cost-effective for non-safety-critical applications.*

3. *Implement the developed techniques using modern CMOS technologies. Develop validation and evaluation mechanisms to comparatively evaluate the proposed techniques with previous ones. Demonstrate the advantages of the proposed techniques in terms of overhead-efficiency and error-tolerance capabilities.*

4. *Incorporate the developed techniques into modern microprocessor pipelines, and achieve a complete and efficient pipeline protection mechanism. Validate the techniques at the system-level by modelling transient fault effects through simulations. Comparatively analyse the efficiencies and the cost of the proposed pipeline protection mechanisms with previous techniques.*

The first objective is fulfilled by the radiation hardened flip-flop architectures, SETTOFF, proposed in Chapter 3. As discussed in Chapter 1 and Chapter 2, the major drawback of most previous techniques is that they are either too expensive, or can only provide a limited level of reliability. The SETTOFF architectures can provide better error-tolerance capabilities than most previous techniques since they offer effective mitigation for both the SEUs occurring in the flip-flop, and the captured SETs originating in the preceding combinational gates. Timing errors can also be naturally addressed by SETTOFF. Meanwhile, SETTOFF minimises the error-tolerance overhead by separating the correction processes for different faults into two levels (circuit-level and architectural-level). The reason for this is that error-corrections normally require much larger overheads than pure error-detection. By investigating the natures of the transient faults, SETTOFF realises circuit-level, on-the-fly corrections for the SEUs occurring during the hold cycle of the flip-flop. This is because these SEUs are most difficult to recover. Other faults, including the captured SETs and SEUs occurring during the write cycle of the flip-flop, can be easily recovered by a pipeline replay operation at the architectural-level. The overhead for incorporating an architectural replay feature can be rather cheap since it normally already exists in modern microprocessors to support speculative operations.

Two versions of the SETTOFF architecture, SETTOFF1 and SETTOFF2, are developed and implemented in both 65nm and 120nm technologies for evaluation. The second version, SETTOFF2, provides a similar level of reliability as SETTOFF1, but can further reduce the implementation overheads noticeably. A novel reliability metric called the transient fault failure rate is proposed and used for evaluating the reliability of the SETTOFFs. Both SETTOFF1 and SETTOFF2 are compared with previous radiation-hardened techniques in terms of their implementation overheads and error-tolerance capabilities. The evaluation results show that the SETTOFFs can effectively reduce both the SEU and SET failure rates to 0, while most previous techniques cannot address both SEUs and SETs. The implementation overheads of the SETTOFFs in terms of area, power, and delay, are smaller or comparable to most previous techniques. Compared to the conventional TMR latch, for instance, SETTOFF2 requires over 50% and 85% less

area and power consumption overheads, respectively. The delay overhead is also reduced by 85%. The results demonstrate that the SETTOFF techniques address the limitations of most previous techniques, and can potentially be used for providing effective and low-cost protection for general logic in non-safety-critical applications. Therefore, the first objective is achieved.

The second objective is achieved by the self-checking technique based on the SETTOFF architecture proposed in Chapter 4. The self-checking technique further improves the reliability of the SETTOFF techniques by allowing SETTOFFs to mitigate not only the SETs and SEUs in the original circuitry, but also errors in the added redundancies. It overcomes the drawback of most previous techniques which cannot address the errors arising in the redundant circuity. The self-checking capability is realised by using a self-checker, which can be shared by multiple SETTOFFs in a register architecture. The sharing of the self-checker minimises the overheads the self-checking feature incurs. The SETTOFF-based self-checking techniques were implemented in 65nm for evaluation and validation. The results show that the self-checker can effectively protect those redundancies that are vulnerable to the soft errors. The extra overhead introduced by the self-checking feature is also insignificant, compared to a pure SETTOFF architecture. In a SETTOFF2-based self-checking technique, for instance, the added self-checking feature only increases the area and power consumption by 28% and 16%, respectively, compared to a single SETTOFF2 architecture. This allows the proposed technique to remain cost-effective compared to most previous techniques.

In order to further validate the efficiencies of the self-checking capability, the technique is also incorporated into the a register file architecture to achieve error-mitigation. The register file protected by the proposed technique was implemented in 65nm technology, and is compared with a register file protected by the traditional ECC technique. The failure rate model proposed in Chapter 3 is adapted to statistically analyse the reliability of the register-level implementations. The results show that the proposed register file produces much lower SET and SEU failure rates for both SBUs and MBUs affecting the RF. Meanwhile, the proposed RF reduces the power overhead by nearly 50%, compared to the ECC-protected RF. The performance overhead is also significantly reduced since most SEUs are corrected on the fly, hence no extra cycles are required for recovering these faults.

The third objective is achieved by the following: The SETTOFF architectures proposed in Chapter 3 and the self-checking architectures proposed in Chapter 4 are both implemented in 65nm and 120nm technologies for evaluation and validation. SPICE simulations were carried out to measure the power and propagation delay of the hardened cells, which were then compared with other previous techniques. The error-tolerance capabilities of the proposed architectures were validated using the transistor-level transient fault injection technique, which is achieved by using independent current sources to simulate the collected charge induced by the particle strikes at circuit nodes. In addition, a novel

failure rate model is developed which can quantitatively analyse and compare the reliability of various designs. The reliability evaluation results validated the error-tolerance capabilities of the proposed designs, and demonstrated that the proposed techniques can provide a higher level of reliability than most previous techniques. Meanwhile, the measurements of implementation overheads reveal that the proposed techniques require less or comparable overheads than the previous techniques.

The final objective is achieved by the complete pipeline protection mechanism realised on an OpenRISC microprocessor proposed in Chapter 5. The pipeline protection is achieved by incorporating the SETTOFF-based self-checking cells into the most vulnerable sequential gates of the pipeline. A pipeline replay recovery mechanism is also developed at the architectural level to recover the corresponding errors detected by the hardened cells. The entire pipeline is protected, the errors occurring in each stages of the pipeline are addressed by the radiation hardened cells in the corresponding stages. The proposed robust OpenRISC microprocessor is implemented in 65nm technology for validation and evaluation. A gate-level transient fault injection and simulation technique is developed, which can automatically inject SETs and SEUs into different parts of the implemented pipeline, and record errors caused by the injected faults. The fault simulation results show that the proposed processor pipeline is robust to both SEUs and SETs occurring in different pipeline stages.

The major limitation of the previous pipeline protection techniques is that they can only protect either the speculative pipeline registers or the RF. The limited error-tolerance capabilities or the unacceptable overheads have made most previous techniques unsuitable for protecting the entire pipeline. The proposed technique, however, can effectively protect both the speculative pipeline registers, and the non-speculative registers storing the intermediate execution results, such as the RF and the flag registers. Compared to the previous pipeline register protection techniques, such as Razor and SEM/STEM, the proposed pipeline register protection requires less or comparable overheads in terms of area, power, and performance. Compare to the conventional ECC-based RF-protection techniques, the proposed RF-protection can provide a noticeable better mitigation capability for SETs and MBUs. Meanwhile, the proposed RF protection requires significantly less power and delay overheads compared to the ECC-based RF-protection.

The three main contributions presented in this thesis provide novel and cost-effective solutions for radiation-induced soft errors in general logic. The reliability that the proposed mechanisms can provide and the low overheads that they require make them suitable for protecting non-safety-critical commercial applications, such as modern microprocessors. The conclusions drawn in this thesis are supported by comprehensive analysis of the proposed techniques implemented on modern CMOS technologies. Novel reliability metrics, fault injection and simulation techniques are especially developed for evaluating the proposed techniques. State-of-art EDA tools are used to help with the circuit implementation and validation processes. It is hoped that the cost-effective

soft error solutions proposed in this thesis can make useful contributions towards the development of future reliable electronic products.

6.2 Future Work

Based on the findings presented in this thesis, several directions for future research are identified and described as follows:

6.2.1 Performance and Power-Efficiency Enhancement for Reliable Systems

Balancing the trade-offs between performance, power consumption, and reliability is a vital task in microprocessor designs. This thesis focuses on effectively improving the reliability of microprocessors by sacrificing minimum power and performance overheads. The primary idea of the proposed techniques is to mitigate soft errors, but as discussed in Chapter 3, timing errors are also naturally addressed. The natural timing error-tolerance capability give us the potential of applying techniques such as over-clocking and Dynamic Voltage Scaling (DVS) to enhance the performance and power efficiency of the proposed radiation-hardened processor. The over-clocking technique, for instance, can effectively improve the system performance but with the danger of inducing timing errors at a certain rate. If that timing error rate can be tolerated by the timing error-tolerance capability of the proposed radiation-hardened design, the system performance can be enhanced without adding massive extra features. Similarly, aggressive DVS is an effective way of reducing system energy consumption, but may also incur timing errors since the speed of the logic gates is reduced due to lower supply voltages. The timing error-tolerance capability can guarantee correct system operations in the presence of timing errors, therefore it is possible to achieve reliable DVS in the proposed robust processor architectures.

In order to fully realise timing error mitigation in the proposed design, the architectural replay recovery mechanism, introduced in Chapter 5, needs to be amended to support timing error recovery. This is because timing errors may repeat in a simple pipeline re-execution, certain voltage or frequency adjustments may be required during the replay for recovering timing errors. This future research direction explores the potential of the proposed radiation-hardened techniques, and can potentially realise simultaneous improvements for performance, power-efficiency, and reliability of microprocessors, which would be a promising contribution towards highly reliable, yet competitive microprocessor designs.

6.2.2 Reliable System Design Automation

It has been discussed in Chapter 1 that as technology scales, reliability issues caused by radiation-induced soft-errors have become a major cause of concern. Certain error-tolerance features have become essential, even for non-safety-critical commercial products, to satisfy reliability requirements. The ultimate goal of this research is to allow designers to achieve the conflicting requirements of area, power dissipations, performance, and reliability in modern microprocessor designs. It is therefore essential to provide designers with the data of these metrics at early design stages, such that designers can select appropriate error-mitigation features to incorporate into certain vulnerable parts of the system. A much more efficient way of doing this is to let the EDA tools automate this process according to the constraints of all the metrics. In order to achieve this, the reliability metric with regards to soft errors needs to be built into EDA tools, and qualitative analysis and estimation of system reliability at early design stage is also crucial. The MTTF and FIT metrics introduced in Chapter 1 can be used to model the reliability of the components comprising the system. However, in order to model the reliability of the whole system, the vulnerability factors by which the intrinsic FIT rate can be derated are also required. The reliability metric, the transient fault failure rate, proposed in Chapter 3, Section 3.6 can therefore be a good starting point to achieve this, by quantifying the transient fault failure rates of various design structures. The error-tolerance architectures proposed in this thesis also need to be fully incorporated into commercial CMOS technologies, and need to be recognised by EDA tools during the design process. This potential future work can provide an efficient and reliable way of designing future reliable systems.

Appendix A

Soft Error Analysis Model

A.0.3 The testbench for the SET injection and simulation

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use std.textio.all;
4  use work.fault_inject.all;
5  use ieee.numeric_std.all;
6  use ieee.math_real.all;
7
8  entity test_orpsoc is end entity test_orpsoc;
9
10 architecture fileio of test_orpsoc is
11
12  --declare text file
13    file faults : text;
14    file memory : text;
15    file fault1 : text;
16    file summary : text;
17
18  --declare orpsoc_top component
19  component orpsoc_top is
20       port (tdo_pad_o : out std_logic;
21         tms_pad_i, tck_pad_i, tdi_pad_i: in std_logic;
22         uart0_srx_pad_i : in std_logic;
23         uart0_stx_pad_o : out std_logic;
24         clk_pad_i, rst_n_pad_i : in std_logic;
25         dump_errorfree : in std_logic;
26         dump_error : in std_logic;
27         error_trig : in std_logic
28        );
29   end component orpsoc_top;
30
31  --declare random number generator component
32   component pseudo_random_number is
33     port(
34      ld : in std_logic;
35      clk : in std_logic;
36      result : out integer;
37      result1 : out integer
38      );
```

```
39   end component pseudo_random_number;
40
41  --orpsoc
42   --Clk and Rst
43   signal clk_pad_i_s :  std_logic := '0';
44   signal rst_n_pad_i_s :  std_logic := '1';
45   --JTAG_DEBUG
46   signal tck_pad_i, tms_pad_i, tdi_pad_i : std_logic;
47   signal tdo_pad_o : std_logic;
48   --UART0
49   signal uart0_srx_pad_i : std_logic;
50   signal uart0_stx_pad_o : std_logic;
51
52  --random generator
53   signal rst :  std_logic := '0';
54   signal result_s : integer;
55   signal result1_s : integer;
56
57  --dump mem control
58   signal dump_errorfree_s : std_logic := '0';
59   signal dump_error_s : std_logic := '0';
60   signal error_trig_s : in std_logic := '0';
61
62 begin
63
64 --instantiation for OPRSoc
65 s1: orpsoc_top port map (tdo_pad_o, tms_pad_i, tck_pad_i, tdi_pad_i,
      uart0_srx_pad_i, uart0_stx_pad_o, clk_pad_i_s, rst_n_pad_i_s,
      dump_errorfree_s, dump_error_s, error_trig);
66
67 --instantiation for a random number generator
68 r1: pseudo_random_number port map (rst, clk_pad_i_s, result_s, result1_s);
69
70     clk_pad_i_s <= not clk_pad_i_s after 10 ns;  ---system clock
71
72 p1: process is
73       variable head_ptr : fault_ptr := null;
74       variable inject_time, inject_period : time;
75       variable ILine1, ILine2, Oline, Oline1, fname : Line;
76       variable FFree, FInject : bit_vector(31 downto 0);
77       variable fc, fd : natural := 0;
78    begin
79
80     -- fault free simulation
81      --do all the rst
82      rst_n_pad_i_s <= '0' after 660 ns, '1' after 1300 ns;  -- system reset
83      rst <= '1' after 660 ns, '0' after 1300 ns;  -- reset for the random
      number generator
84      dump_errorfree_s <= '1' after 2592920 ns, '0' after 2592930 ns;  -- this
      is the point for dumping out the data memory after fault free simulation
85      wait for 2700000 ns;  ----------this is the program execution time
86
87     -- sequential fault simulation
88      rst_n_pad_i_s <= '0' after 660 ns, '1' after 1300 ns;
89      head_ptr := first_fault;
90         file_open (faults, faults.txt, write_mode);
91     while head_ptr /= null loop
92         head_ptr.simulating := true;
```

```
93          dump_error_s <= '1' after 2592920 ns, '0' after 2592930 ns;  -- this
        is the point for dumping out the data memory during fault simulations
94          fc := fc + 1;  -- fault count
95          inject_time := result_s * 27000 ns; ----------------this is the program
        execution time divided by 100, since result_s is a random number between 0
         to 100.
96          wait for inject_time;  -- a random number between 0 and programm time
97          error_trig_s <= '1';
98          inject_period := (result1_s + 1) * 2 ns; -- random SET widths
99          wait for inject_period;   -- inject SET with particular widths
100         error_trig_s <= '0';
101       wait for (2700000 ns - inject_time - inject_period);
102
103       -- compare results with free case and print differences time
104       --wait for 5 ns;
105         file_open (memory, memory.txt, read_mode);
106         file_open (fault1, fault1.txt, read_mode);
107
108       while not endfile(memory) loop
109         readline(memory, ILine1);
110         read(ILine1, FFree);
111         readline(fault1, ILine2);
112         read(ILine2, FInject);
113         if FFree /= FInject then
114           head_ptr.detected := true;
115           fname := new string'(head_ptr.fault_name.all);
116           writeline(faults, fname);
117           write(OLine, string'(Fault #), left, 0);
118           write(OLine, fc, left, 5);
119           write(OLine, string'(Detected by mem:), left, 0);
120           write(OLine, FInject, right, 9);
121           write(OLine, string'( expected: ), left, 0);
122           write(OLine, FFree, right, 9);
123           writeline(faults, OLine);
124         end if;
125       end loop;
126         file_close(memory);
127         file_close(fault1);
128         -- reset the system
129         rst_n_pad_i_s <= '0' after 660 ns, '1' after 1300 ns;
130         head_ptr.simulating := false;
131         head_ptr := head_ptr.next_fault;
132   end loop;
133         file_close(faults);
134
135         -- summarize results
136       file_open (summary, summary.txt, write_mode);
137       head_ptr:=first_fault;
138       while head_ptr /= null loop
139         if head_ptr.detected then
140           fd := fd + 1;
141         end if;
142         head_ptr := head_ptr.next_fault;
143       end loop;
144       write(OLine1, string'(Fault Cover:), left, 0);
145       writeline(summary, OLine1);
146       write(OLine1, fc, right, 8);
147       write(OLine1, string'( faults,), left, 0);
148       write(OLine1, fd, right, 8);
```

```
149         write(OLine1, string'( detected), left, 0);
150         writeline(summary, OLine1);
151
152     wait; -- halt
153 end process p1;
154
155 end architecture fileio;
```

References

[1] G. E. Moore, "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff." *Solid-State Circuits Society Newsletter, IEEE*, vol. 11, no. 5, pp. 33–35, Sept 2006.

[2] T. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *Electron Devices, IEEE Transactions on*, vol. 26, no. 1, pp. 2–9, Jan 1979.

[3] J. F. Ziegler and H. Puchner, *SER - History, Trends and Challenges.* Cypress Semiconductor Corporation, 2004.

[4] S. Mukherjee, *Architecture Design for Soft Errors.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[5] J. Ziegler and W. Lanford, "The effect of sea level cosmic rays on electronic devices," in *Solid-State Circuits Conference. Digest of Technical Papers. 1980 IEEE INT*, vol. XXIII, Feb. 1980, pp. 70 – 71.

[6] M. Baze and S. Buchner, "Attenuation of single event induced pulses in cmos combinational logic," *Nuclear Science, IEEE Transactions on*, vol. 44, no. 6, pp. 2217–2223, Dec 1997.

[7] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson, "On latching probability of particle induced transients in combinational networks," in *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, June 1994, pp. 340–349.

[8] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 389 – 398.

[9] P. Hazucha and C. Svensson, "Impact of cmos technology scaling on the atmospheric neutron soft error rate," *Nuclear Science, IEEE Transactions on*, vol. 47, no. 6, pp. 2586–2594, Dec 2000.

[10] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, Jul 1999.

[11] N. Mahatme, N. Gaspard, S. Jagannathan, T. Loveless, B. Bhuva, W. Robinson, L. Massengill, S.-J. Wen, and R. Wong, "Impact of supply voltage and frequency on the soft error rate of logic circuits," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 6, pp. 4200–4206, Dec 2013.

[12] C. Slayman, "Soft error trends and mitigation techniques in memory devices," in *Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual*, Jan 2011, pp. 1–5.

[13] O. Musseau, F. Gardic, P. Roche, T. Corbiere, R. Reed, S. Buchner, P. McDonald, J. Melinger, L. Tran, and A. Campbell, "Analysis of multiple bit upsets (mbu) in cmos sram," *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, pp. 2879–2888, Dec 1996.

[14] W. Bennett, N. Hooten, R. Schrimpf, R. Reed, R. Weller, M. Mendenhall, A. Witulski, and D. Wilkes, "Experimental characterization of radiation-induced charge sharing," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 6, pp. 4159–4165, Dec 2013.

[15] N. Seifert, B. Gill, V. Zia, M. Zhang, and V. Ambrose, "On the Scalability of Redundancy based SER Mitigation Schemes," in *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, 2007, pp. 1–9.

[16] F. Ruckerbauer and G. Georgakos, "Soft error rates in 65nm srams–analysis of new phenomena," in *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, 2007, pp. 203–204.

[17] C. Zhao, X. Bai, and S. Dey, "Evaluating transient error effects in digital nanometer circuits," *Reliability, IEEE Transactions on*, vol. 56, no. 3, pp. 381–391, Sept 2007.

[18] T. Merelle, F. Saigne, B. Sagnes, G. Gasiot, P. Roche, T. Carriere, M.-C. Palau, F. Wrobel, and J. M. Palau, "Monte-carlo simulations to quantify neutron-induced multiple bit upsets in advanced srams," *Nuclear Science, IEEE Transactions on*, vol. 52, no. 5, pp. 1538–1544, Oct 2005.

[19] M. Raine, G. Hubert, M. Gaillardin, P. Paillet, and A. Bournel, "Monte carlo prediction of heavy ion induced mbu sensitivity for soi srams using radial ionization profile," *Nuclear Science, IEEE Transactions on*, vol. 58, no. 6, pp. 2607–2613, Dec 2011.

[20] R. Pawlowski, J. Crop, M. Cho, J. Tschanz, V. De, S. Borkar, T. Fairbanks, H. Quinn, and P. Chiang, "A reference design for effective characterization of soft error vulnerability of ultra-low voltage logic and memory circuits," in *Silicon Errors in Logic - System Effects workshop, SELSE 2014* , April 2014.

[21] H. Fuketa, R. Harada, M. Hashimoto, and T. Onoye, "Measurement and analysis of alpha-particle-induced soft errors and multiple-cell upsets in 10t subthreshold sram," *Device and Materials Reliability, IEEE Transactions on*, vol. 14, no. 1, pp. 463–470, March 2014.

[22] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron cmos technology," *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, pp. 2874 –2878, Dec. 1996.

[23] E. Rotenberg, "Ar-smt: a microarchitectural approach to fault tolerance in microprocessors," in *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, June 1999, pp. 84 –91.

[24] M. Ebrahimi, A. Evans, M. Tahoori, R. Seyyedi, E. Costenaro, and D. Alexandrescu, "Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–6.

[25] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 124 –134, Mar. 1984.

[26] M. Abramovici, M. A. Breuer, and F. A.D., *Digital Systems Testing and Testable Design*. Wiley-IEEE Press, 1990.

[27] W. Van Gils, "A triple modular redundancy technique providing multiple-bit error protection without using extra redundancy," *Computers, IEEE Transactions on*, vol. C-35, no. 7, pp. 623 –631, July 1986.

[28] M. Favalli and C. Metra, "TMR voting in the presence of crosstalk faults at the voter inputs," *Reliability, IEEE Transactions on*, vol. 53, no. 3, pp. 342 – 348, Sept. 2004.

[29] L. Sterpone and M. Violante, "Analysis of the robustness of the tmr architecture in sram-based fpgas," *Nuclear Science, IEEE Transactions on*, vol. 52, no. 5, pp. 1545 – 1549, Oct. 2005.

[30] D. Rennie, D. Li, M. Sachdev, B. Bhuva, S. Jagannathan, S. Wen, and R. Wong, "Performance, metastability, and soft-error robustness trade-offs for flip-flops in 40 nm cmos," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 8, pp. 1626–1634, Aug 2012.

[31] S. Shambhulingaiah, S. Chellappa, S. Kumar, and L. Clark, "Methodology to optimize critical node separation in hardened flip-flops," in *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, March 2014, pp. 486–493.

[32] M. Fazeli, A. Patooghy, S.-G. Miremadi, and A. Ejlali, "Feedback redundancy: A power efficient seu-tolerant latch design for deep sub-micron technologies," in *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, June 2007, pp. 276–285.

[33] M. Fazeli, S.-G. Miremadi, A. Ejlali, and A. Patooghy, "Low energy single event upset/single event transient-tolerant latch for deep submicron technologies," *Computers Digital Techniques, IET*, vol. 3, no. 3, pp. 289–303, 2009.

[34] L. Anghel, V. Savulimedu, D. Alexandrescu, A. Steininger, K. Schneider-Hornstein, and E. Costenaro, "Single event effects in muller c-elements and asynchronous circuits over a wide energy spectrum," in *Silicon Errors in Logic - System Effects workshop, SELSE 2014* , April 2014.

[35] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[36] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 86 –94.

[37] L. Anghel and M. Nicolaidis, "Cost reduction and evaluation of a temporary faults detecting technique," in *Design, Automation and Test in Europe Conference and Exhibition. Proceedings*, 2000, pp. 591 –598.

[38] G. Bany Hamad, S. Hasan, O. Mohamed, and Y. Savaria, "New insights into the single event transient propagation through static and tspc logic," *Nuclear Science, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[39] H. Yu, M. Nicolaidis, and L. Anghel, "An effective approach to detect logic soft errors in digital circuits based on graal," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, March 2009, pp. 236 –240.

[40] M. Nicolaidis, "Graal: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies," in *Test Conference, 2007. ITC 2007. IEEE International*, Oct. 2007, pp. 1 –10.

[41] N. Avirneni and A. Somani, "Low overhead soft error mitigation techniques for high-performance and aggressive designs," *Computers, IEEE Transactions on*, vol. 61, no. 4, pp. 488 –501, Apr. 2012.

[42] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec. 2003, pp. 7 – 18.

[43] S. Das, S. Pant, D. Roberts, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "A self-tuning dvs processor using delay-error detection and correction," in *VLSI Circuits, 2005. Digest of Technical Papers. 2005 Symposium on*, June 2005, pp. 258–261.

[44] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "RazorII: In situ error detection and correction for PVT and SER tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 32 –48, Jan. 2009.

[45] M. Nicolaidis, R. Perez, and D. Alexandrescu, "Low-cost highly-robust hardened cells using blocking feedback transistors," in *VLSI Test Symposium, 2008. VTS 2008. 26th IEEE*, April 2008, pp. 371–376.

[46] S. Lin, Y.-B. Kim, and F. Lombardi, "A 11-transistor nanoscale cmos memory cell for hardening to soft errors," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 900 –904, May 2011.

[47] Y. Sasaki, K. Namba, and H. Ito, "Soft error masking circuit and latch using schmitt trigger circuit," in *Defect and Fault Tolerance in VLSI Systems, 2006. DFT '06. 21st IEEE International Symposium on*, Oct. 2006, pp. 327 –335.

[48] S. Lin, Y.-B. Kim, and F. Lombardi, "Design and performance evaluation of radiation hardened latches for nanoscale cmos," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 7, pp. 1315 –1319, July 2011.

[49] M. Glorieux, S. Clerc, G. Gasiot, J.-L. Autran, and P. Roche, "New d-flip-flop design in 65nm cmos for improved seu and low power overhead at system level," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 6, pp. 4381–4386, Dec 2013.

[50] S. Lin, Y.-B. Kim, and F. Lombardi, "Analysis and design of nanoscale cmos storage elements for single-event hardening with multiple-node upset," *Device and Materials Reliability, IEEE Transactions on*, vol. 12, no. 1, pp. 68 –77, March 2012.

[51] S. Kim and A. Somani, "Area efficient architectures for information integrity in cache memories," in *Computer Architecture, 1999. Proceedings of the 26th International Symposium on*, 1999, pp. 246 –255.

[52] J. Guo, L. Xiao, and Z. Mao, "Novel low-power and highly reliable radiation hardened memory cell for 65 nm cmos technology," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, no. 7, pp. 1994–2001, July 2014.

[53] S. Kim and A. Somani, "An adaptive write error detection technique in on-chip caches of multi-level caching systems," *Microprocessors and Microsystems*, vol. 22, no. 9, pp. 561 – 570, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933198001094

[54] C.-H. Chen and A. Somani, "Fault-containment in cache memories for tmr redundant processor systems," *Computers, IEEE Transactions on*, vol. 48, no. 4, pp. 386 –397, Apr 1999.

[55] T. Austin, "Diva: a reliable substrate for deep submicron microarchitecture design," in *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, 1999, pp. 196 –207.

[56] J. A. Blome, S. Gupta, S. Feng, and S. Mahlke, "Cost-efficient soft error protection for embedded microprocessors," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES '06, 2006, pp. 421–431.

[57] P. Montesinos, W. Liu, and J. Torrellas, "Using register lifetime predictions to protect register files against soft errors," in *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, 2007, pp. 286–296.

[58] ——, "Shield: Cost-effective soft-error protection for register files."

[59] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, Dec. 2003, pp. 29 – 40.

[60] J. Yan and W. Zhang, "Compiler-guided register reliability improvement against soft errors," in *In International Conference on Embedded Software*, 2005, pp. 203–209.

[61] G. Memik, M. Kandemir, and O. Ozturk, "Increasing register file immunity to transient errors," in *Design, Automation and Test in Europe, 2005. Proceedings*, March 2005, pp. 586 – 591 Vol. 1.

[62] M. Fazeli, A. Namazi, and S. Miremadi, "Robust register caching: An energy-efficient circuit-level technique to combat soft errors in embedded processors," *Device and Materials Reliability, IEEE Transactions on*, vol. 10, no. 2, pp. 208 –221, June 2010.

[63] C. Weaver, J. Emer, S. Mukherjee, and S. Reinhardt, "Reducing the soft-error rate of a high-performance microprocessor," *Micro, IEEE*, vol. 24, no. 6, pp. 30 –37, nov.-dec. 2004.

[64] W.-M. W. Hwu, S. A. Mahlke, W. Y. Chen, P. P. Chang, N. J. Warter, R. A. Bringmann, R. G. Ouellette, R. E. Hank, T. Kiyohara, G. E. Haab, J. G. Holm, and D. M. Lavery, "The superblock: An effective technique for vliw and superscalar compilation," *THE JOURNAL OF SUPERCOMPUTING*, vol. 7, pp. 229–248, 1993.

[65] S. Mahlke, D. Lin, W. Chen, R. Hank, and R. Bringmann, "Effective compiler support for predicated execution using the hyperblock," in *Microarchitecture, 1992. MICRO 25., Proceedings of the 25th Annual International Symposium on*, dec 1992, pp. 45 –54.

[66] W. Zhang, N. Vijaykrishnan, M. Kandemir, M. Irwin, D. Duarte, and Y.-F. Tsai, "Exploiting vliw schedule slacks for dynamic and leakage energy reduction," in *Microarchitecture, 2001. MICRO-34. Proceedings. 34th ACM/IEEE International Symposium on*, Dec. 2001, pp. 102 – 113.

[67] M. Nicolaidis, "Design for soft error mitigation," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 405 – 418, Sept. 2005.

[68] ——, "Electronic circuit assembly comprising at least one memory with error correcting means," 2001.

[69] P. Oikonomakos and M. Zwolinski, "On the design of self-checking controllers with datapath interactions," *Computers, IEEE Transactions on*, vol. 55, no. 11, pp. 1423 –1434, Nov. 2006.

[70] N. Bowen and D. Pradham, "Processor- and memory-based checkpoint and rollback recovery," *Computer*, vol. 26, no. 2, pp. 22–31, Feb 1993.

[71] S. Nassif, K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, E. Nowak, D. Pearson, and N. Rohrer, "High performance cmos variability in the 65nm regime and beyond," in *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, 2007, pp. 569–571.

[72] K. Agarwal, M. Agarwal, D. Sylvester, and D. Blaauw, "Statistical interconnect metrics for physical-design optimization," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1273–1288, 2006.

[73] K. Agarwal, D. Sylvester, D. Blaauw, T. Liu, S. Nassif, and S. Vrudhula, "Variational delay metrics for interconnect timing analysis," in *Design Automation Conference, 2004. Proceedings. 41st*, 2004, pp. 381–384.

[74] L. Zhang, Y. Hu, and C. Chen, "Statistical timing analysis in sequential circuit for on-chip global interconnect pipelining," in *Design Automation Conference, 2004. Proceedings. 41st*, July 2004, pp. 904–907.

[75] J. Xu, A. Roy, and M. Chowdhury, "Analysis of power consumption and ber of flip-flop based interconnect pipelining," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[76] D. Hong, C.-K. Ong, and K.-T. Cheng, "Bit-error-rate estimation for high-speed serial links," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 12, pp. 2616–2627, Dec 2006.

[77] J. A. Gubner, *Probability and Random Processes for Electrical and Computer Engineers*. Cambridge: Cambridge Univ. Press, 2006.

[78] R. Harada, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "SET pulse-width measurement eliminating pulse-width modulation and within-die process variation effects," in *Reliability Physics Symposium (IRPS), 2012 IEEE International*, 2012, pp. SE.1.1–SE.1.6.

[79] ——, "Measurement circuits for acquiring SET pulsewidth distribution with sub-FO1-inverter-delay resolution," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, 2010, pp. 839–844.

[80] T. Loveless, J. Kauppila, S. Jagannathan, D. Ball, J. Rowe, N. Gaspard, N. Atkinson, R. Blaine, T. Reece, J. Ahlbin, T. Haeffner, M. Alles, W. Holman, B. Bhuva, and L. Massengill, "On-chip measurement of single-event transients in a 45 nm silicon-on-insulator technology," *Nuclear Science, IEEE Transactions on*, vol. 59, no. 6, pp. 2748–2755, Dec 2012.

[81] STMicroelectronics, "CORE65LPLVT_1.20V 4.1 Standard Cell Library User Manual and Databook," June 2006.

[82] A. Ejlali, B. Al-Hashimi, M. Schmitz, P. Rosinger, and S.-G. Miremadi, "Combined time and information redundancy for seu-tolerance in energy-efficient real-time systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 323–335, April 2006.

[83] A. Narasimhan and R. Sridhar, "Impact of variability on clock skew in h-tree clock networks," in *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, 2007, pp. 458–466.

[84] OpenCores, "OpenRISC 1000 architecture manual," Apr. 2006. [Online]. Available: http://opencores.org/openrisc,overview

[85] D. Lampret, "OpenRISC 1200 IP core specification," 2011. [Online]. Available: http://opencores.org/openrisc,downloads

[86] N. Mehdizadeh, M. Shokrolah-Shirazi, and S. Miremadi, "Analyzing fault effects in the 32-bit openrisc 1200 microprocessor," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, Mar. 2008, pp. 648 –652.

[87] M. Zwolinski, "A technique for transparent fault injection and simulation," *Microelectronics Reliability*, pp. 797–804, 2000.

[88] ——, *Digital System Design with VHDL*. Prentice Hall, 2003.

[89] J. Baxter, "ORPSoC - OpenRISC Reference Platform SoC." [Online]. Available: http://opencores.org/openrisc,orpsocv2

[90] Synopsys, "Liberty ncx user guide," June 2012.