UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Error Resilient Techniques for Storage Elements of Low Power Design

by

Sheng Yang

Thesis for the degree of Doctor of Philosophy

July 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Doctor of Philosophy

ERROR RESILIENT TECHNIQUES FOR STORAGE ELEMENTS OF LOW POWER DESIGN

by Sheng Yang

Over two decades of research has led to numerous low-power design techniques being reported. Two popular techniques are supply voltage scaling and power gating. This thesis studies the impact of these two design techniques on the reliability of embedded processor registers and memory systems in the presence of transient faults; and with the aim to develop and validate efficient mitigation techniques to improve reliability with small cost of energy consumption, performance and area overhead.

This thesis presents three original contributions. The first contribution presents a technique for improving the reliability of embedded processors. A key feature of the technique is low cost, which is achieved through reuse of the scan chain for state monitoring, and it is effective because it can correct single and multiple bit errors through hardware and software respectively. To validate the technique, ARM® Cortex™-M0 embedded microprocessor is implemented in FPGA and further synthesised using 65-nm technology to quantify the cost in terms of area, latency and energy. It is shown that the presented technique has a small area overhead (8.6%) with less than 4% worst-case increase in critical path. The second contribution demonstrates that state integrity of flip-flops is sensitive to process, voltage and temperature (PVT) variation through measurements from 82 test chips. A PVT-aware state protection technique is presented to ensure state integrity of flip-flops while achieving maximum leakage savings. The technique consists of characterisation algorithm and employs horizontal and vertical parity for error detection and correction. Silicon results show that flip-flops state integrity is preserved while achieving up to 17.6% reduction in retention voltage across 82-dies. Embedded processors memory systems are susceptible to transient errors and blanket protection of every part of memory system through ECC is not cost effective. The final contribution addresses the reliability of embedded processor memory systems and describes an architectural simulation-based framework for joint optimisation of reliability, energy consumption and performance. Accurate estimation of memory reliability with targeted protection is proposed to identify and protect the most vulnerable part of the memory system to minimise protection cost. Furthermore, L1-cache resizing together with voltage and frequency scaling is proposed for further energy savings while maintaining performance and reliability. The contributions presented are supported by detailed analyses using state-of-the-art design automation tools, in-house software tools and validated using FPGA and silicon implementation of commercial low power embedded processors.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Sheng Yang , declare that the thesis entitled *Error Resilient Techniques for Storage Elements of Low Power Design* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as listed in Chapter 1 Section 1.7

Signed:.....

Date:.....

Acknowledgements

I am indebted to my supervisor professor Bashir M. Al-Hashimi for his supervision and continuous guidance throughout my PhD. I have learnt tremendous amounts from him, and especially, I am inspired by his research vision and devotion to work. This PhD project would not have been possible without his supports and encouragement. I would also like to extend my deepest gratitude to my industrial advisor professor David Flynn for his excellent technical advice and suport, and invaluable discussions on research ideas. My thanks also go to Dr. Saqib Khursheed and Dr. Harry Oldham for their insightful discussions and reviewing of this work.

I would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for supporting my work by means of scholarship, the ARM-ECS Research Center in the School of Electronics and Computer Science, University of Southampton for providing state of the art research facilities. I want to thank ARM Ltd. for allowing me to spend 4 months working at their offices for my Ph.D. My special thanks goes to James Myers, John Biggs, Anand Savanth and Karthik Sivashankar from ARM, and Matthew Swabey from Purdue University, for their technical support in successfully fabricating and testing the silicon chip used in this thesis. I also benefited from the technical discussions with Dr. Shidhartha Das, David Bull, Sachin Idgunji, Dave Howard and Dr. Paul Whatmough during my time in ARM.

I would like to take this opportunity to acknowledge my colleagues for their invaluable support and useful discussions throughout my Ph.D. These people include, but are not limited to: Dr. Rishad Shafik, Dr Geoff Merrett, Dr. Amit Acharyya, Dr. Saqib Khursheed, Dr. Mustafa Imran-Ali, Dr. Zhou Dafeng, Dr. Aissa Melouki, Hamed Shahidipour, Dr. Jatin Mistry, Dr. Zhong Shida, Jedrzej Kufel and Luis Maeda-Nunez.

And finally, I like to offer warmest thanks to my father Shuijin Yang, my mother Ping Zhang, and my girl friend Yunxu Zhang for their love and support, without which it would not have been possible for me to do my research.

Chapter 1

Introduction

Since the invention of integrated circuits in 1958, and driven by the demand for cost reduction and increasing functionality, CMOS transistor technology scaling (including transistor size scaling and supply voltage scaling) has enabled the IC industry to integrate more transistors within the same silicon area. Borkar, et al. [1] showed that each transition to a new generation, reduces transistor size and supply voltage by 30%, doubles the chip density, reduces gate delay by 30%, reduces overall power by 50% and increases leakage power by around 5x. The side effect of technology scaling is the reduction of system reliability. This is because smaller transistors lead to lower critical charge and higher integration level increases system complexity. The other side effect of technology scaling is the increase in power density due to the increase in transistor density. Historically dynamic power has dominated the power consumption of digital integrated circuits and has been the main focus of power reduction. As technology scaling continues, leakage power consumption continues to grow in dominance and leakage power reduction also becomes an important part of modern digital circuits. Both leakage and dynamic power reduction techniques exacerbate reliability problems. This thesis describes new methods to improve the reliability of modern low power digital circuits. These methods have been implemented using standard EDA tools and validated on commercial low power processors.

This chapter gives an overview of reliability and low power design of digital integrated circuits and provides preliminary information for the subsequent thesis chapters. Technology scaling and its major impacts are discussed in Section 1.1. Digital circuit power reduction techniques are discussed in Section 1.2, which includes dynamic power reduction and leakage power reduction techniques. Digital circuit reliability problems are discussed in Section 1.3. Error control coding is an effective technique for ensuring data integrity which can be applied to improve system reliability, which is discussed in Section 1.4. The challenges facing low power reliable design are discussed in Section 1.5. The contribution of each chapter is summarised in Section 1.6 and finally the list of publications generated from the research in this thesis is given in Section 1.7

1.1 Impacts of Technology Scaling

Cost reduction of IC fabrication is the main reason behind technology scaling. For the past 50 years semiconductor technology scaling has reduced transistor size by 30% and doubled the performance every two to three years. Transistor performance is inversely proportional to propagation delay which can be approximated as Equation 1.1 [2]:

$$T_d \propto \frac{C_L \cdot V_{dd}}{V_{dd} - V_{th}} \tag{1.1}$$

where C_L is load capacitance, V_{dd} is supply voltage and V_{th} is threshold voltage. Transistor size scaling reduces load capacitance and leads to shorter propagation delay. However supply voltage scaling increases propagation delay, so to maintain or further reduce propagation delay the threshold voltage has to be scaled down. Transistor dynamic power consumption is described using Equation 1.2 [3]:

$$p_{dynamic} = \alpha \cdot V_{dd}^2 \cdot C_L \tag{1.2}$$

where α is activity factor, C_L is load capacitance and V_{dd} is supply voltage. Dynamic power is proportional to load capacitance and the square of supply voltage. Both load capacitance and supply voltage reduce with technology scaling, and dynamic power reduces with technology scaling. There are two main components of leakage power: gate leakage and sub-threshold leakage. Gate leakage current increases exponentially with reduction in gate oxide thickness [4]; however high-k metal gate reduces the leakage by 2-3 order of magnitude [5, 6]. The most significant contributor to leakage power consumption of current technology generation is sub-threshold leakage current which is calculated by Equation 1.3 [3]:

$$I_{ds} \propto e^{\frac{V_{gs}-V_{th}}{n \cdot V_T}} \cdot (1 - e^{-\frac{V_{ds}}{V_T}}) \tag{1.3}$$

where V_{gs} is gate to source voltage, V_{th} is transistor threshold voltage, V_{ds} is drain to source voltage and V_T is thermal voltage $\frac{KT}{q}$. For $V_{ds} \gg V_T$ $1 - e^{-V_{ds}/V_T} \approx 1$, which means that if V_{ds} is larger than 100-mV then sub-threshold leakage is almost independent of V_{ds}. Sub-threshold leakage can be approximated as:

$$I_{ds} \propto e^{\frac{V_{gs}-V_{th}}{n \cdot V_T}} \tag{1.4}$$

Sub-threshold leakage current increases exponentially with the reduction of threshold voltage V_{th}. Threshold voltage is reduced in order to maintain the performance of

the transistor (Equation 1.1). As the result, sub-threshold leakage current increases exponentially with technology scaling.

Technology scaling also has a number of impacts on system reliability. This is because small transistor size and high integration density increases system soft error rate [7]. As transistor size continues to shrink, process variation in transistor channel length, width, oxide thickness and dopant concentration increases. These variations cause the difference in transistor threshold voltage [8, 9] that leads to the uncertainty in circuit delay of timing paths and performance of fabricated chips. Process variation also exacerbates the reliability problem by reducing the critical charge and noise margin of the storage units. The degradation of reliability and the increases in leakage power are becoming major obstacles for further transistor scaling [10–13].

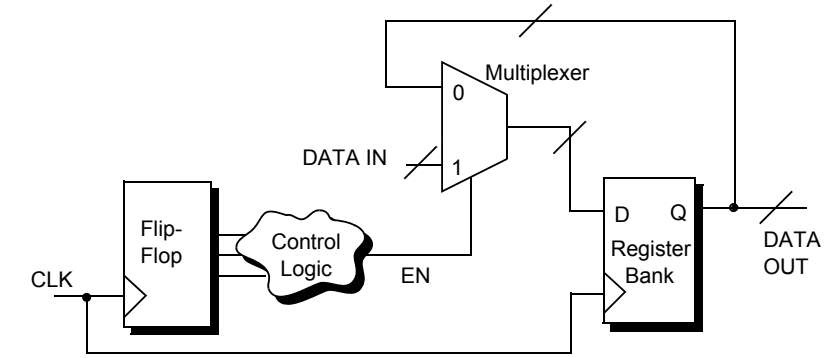
1.2 Power Reduction Techniques

There are two power consumption components in an digital circuit: dynamic power and leakage power. Various power reduction techniques are described in this section, which is divided into dynamic power reduction and leakage power reduction techniques.

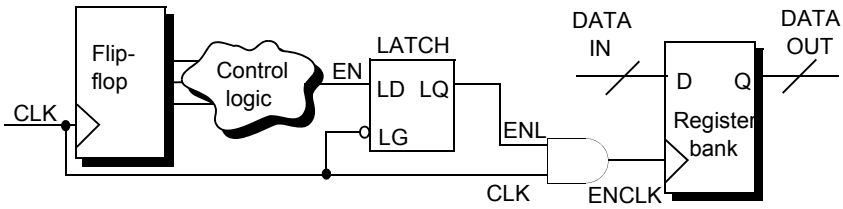
1.2.1 Dynamic Power Reduction

Dynamic power has been the dominant part of digital circuit power consumption for many years. It can be seen from Equation 1.2 that switching activity, load capacitance and supply voltage are three contributing factors of dynamic power. Load capacitance is reduced with transistor size scaling inherently. Dynamic power reduction techniques involve reducing switching activity, reducing supply voltage or both.

In synchronous digital circuits, the clock tree dominates dynamic power consumption due to the constant switching of clock signals and the large fan-out of clock tree. Clock gating can reduce clock tree power consumption [14–18]. Registers are used to store data and states information, whose update is controlled by clock signals. Not all registers require update in every clock cycle; hence clock gating was introduced to stop the switching of parts of clock tree where registers do not need updating. This is achieved by inserting a clock gating cell between clock tree buffers and registers, which allows the clock control signal to enable or disable the register's clock input. Figure 1.1 shows the difference between designs with and without clock gating. Figure 1.1.(a) shows a circuit without clock gating. To enable or disable the updates of an register, a multiplexer is inserted which selects the input between updating new data or recirculating old data stored in the register. Figure 1.1.(b) shows a circuit with clock gating, where a clock gating cell including a latch and an AND gate is used to stop the toggling of the clock signal. When multiple registers are controlled by one enable signal, clock gating can



(a) without clock gating



(b) with clock gating

Figure 1.1: Clock gating architecture [22].

even reduce the chip area. This is because, without clock gating, each register requires one multiplexer, whereas with clock gating, a group of registers can share a single clock gating cell. Various automatic clock gating insertion algorithms [19–21] were developed to improve the effectiveness of clock gating designs, which are well supported by standard EDA (Electronic Design Automation) tools such as the Synopsys Power Compiler.

As the dynamic power is proportional to the square of the supply voltage (Equation 1.2), the most effective way to reduce dynamic power is through supply voltage scaling. When supply voltage is reduced circuit delay is increased (Equation 1.1); however digital circuits do not always operate at their maximum frequency due to workload demands [23]. Dynamic Voltage and Frequency Scaling (DVFS) reduces the supply voltage and operating frequency of circuits under light workload to improve energy efficiency. Various DVFS algorithms [17, 24–29] were developed to maximise energy reduction.

1.2.2 Leakage Power Reduction

Leakage power becomes an increasingly significant part of overall chip power consumption with CMOS technology scaling [30]. There are three types of leakage current in a MOSFET transistor: gate leakage, sub-threshold leakage and junction tunnelling leakage. Gate leakage current increases exponentially with reduction in the gate oxide thickness [4]. With silicon dioxide gate thickness approaching scaling limits there is a

rapid increase in gate leakage current [31, 32]. When the transistor size is scaled below 40nm, with S_iO_2 as the gate dielectric, the gate leakage current density would exceed the current density limit [30]. The threshold voltage of MOSFET transistors is reduced to increase the performance, but the sub-threshold leakage current increases exponentially with the reduction of threshold voltage (equation 1.4). Junction tunnelling leakage is an order of magnitude less than gate leakage and sub-threshold leakage for the recent technology generation [33], therefore it will not be discussed in detail in this work.

A circuit has two states: an active state where it is doing useful work and an idle state where it is not doing any work. Numerous approaches have been proposed to minimise the leakage power, which can be categorised into active circuit leakage power reduction and idle circuit leakage power reduction. The active circuit leakage reduction technique includes high-k metal gate [34], dual-threshold standard cells [35]. The high-K metal gate was proposed [34] to reduce the gate leakage, which has been implemented in technology nodes of 45nm and below [36]. To reduce the sub-threshold leakage while maintaining the performance, dual threshold standard cell synthesis was proposed [35] to reduce the sub-threshold leakage of the gates in the non-critical path by replacing low-Vt high performance transistors with the high-Vt less leaky transistors. These slower high-Vt transistors are not in the critical path so they will not affect the performance of the circuit.

A circuit's leakage power can be further reduced by cutting off or scaling its power supply when the circuit is in an idle state. There are three leakage reduction techniques involved powering off/down a idle circuit. If a chip is idle for a long time and there is no constraint on the wake-up delay, then its external power supply can be turned off completely during sleep mode; in this way the maximum leakage reduction can be achieved. However, the switching of the external power supply takes relatively long time, and some energy and time are wasted in initializing the circuit when it wakes up. If a device has a relatively short idle time and requires fast wake-up, cutting off the chip power supply will reduce its reaction speed. Therefore power gating [37] was proposed to reduce the leakage power of a circuit in sleep mode but maintain a fairly short wake-up latency. In a power-gated circuit, the logic blocks are connected to power supply through high-Vt power transistors shown in Figure 1.2. When the circuit is idle the power transistors are turned off and the leakage power of the logic blocks is limited by those power transistors, it was reported that power gating achieves 95% of leakage power reduction for ARM926EJ^{TM} [38].

Power gating turns off logic and registers, so the states of registers are lost. For energy efficiency the state of registers can be preserved while logic is powered down so-called State Retention Power Gating (SRPG). To support this, a state retention latch [39] can be added which is kept powered during power gating of the rest of the circuit. Figure 1.3 shows a specially designed flip-flop incorporating state retention latch. The master flip-flop is connected to Vdd through a power transistor and the slave retention latch is kept

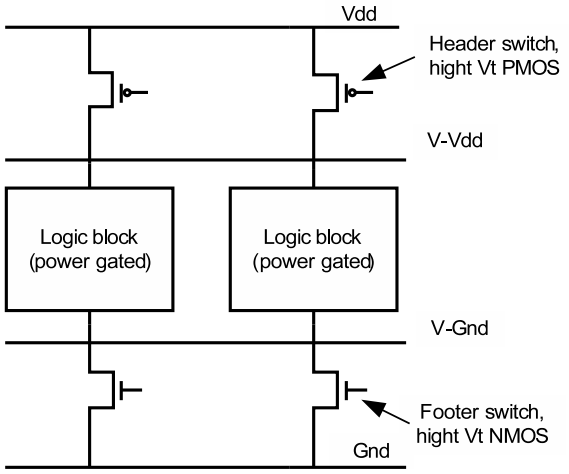


Figure 1.2: Power gating design architecture

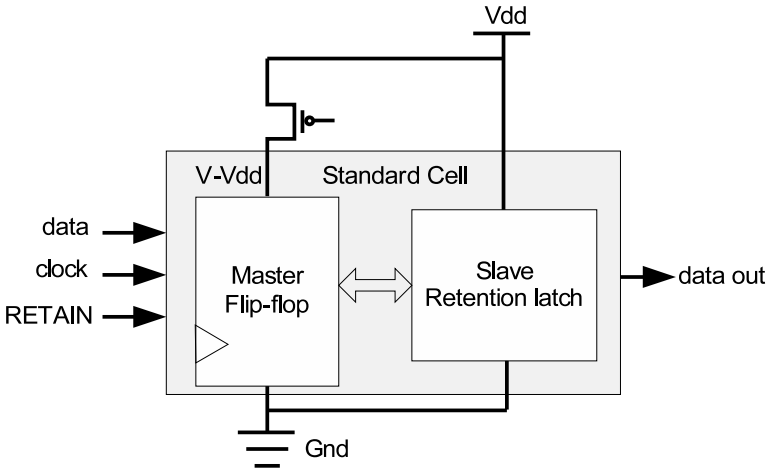


Figure 1.3: State Retention Flip-Flop

always-on. The master flip-flop consists of low-Vt transistors for fast switching during active mode. The slave retention latch consists of high-Vt transistors for low leakage during sleep mode. In addition to the data and clock input the state retention flip-flop has a control signal, RETAIN. When the power-gated circuit is switched to sleep mode, RETAIN is set to '1' to transfer data from the master flip-flop to the slave retention latch, and before the power-gated circuit is switched to active mode, RETAIN is set to '0' to restore data back to the master flip-flop.

The other idle circuit leakage power reduction technique is voltage scaling. The supply voltage of idle circuit can be scaled down aggressively to the point where the storage cells still retain their data [40–42]. The architecture of scaled voltage state retention is shown in Figure 1.4. In active mode signal 'Sleep' is 0, sleep transistor M1 is on and M2 is off, and the logic block is connected to nominal Vdd and clocked with full speed. In sleep mode signal 'Sleep' is 1, sleep transistor M1 is off and M2 is on, and the logic block is connected to scaled Vdd to achieve leakage saving. This technique does not require

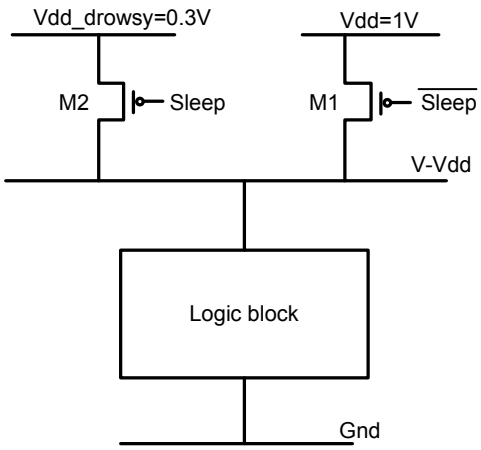


Figure 1.4: Scaled voltage leakage reduction

additional state retention latches and has a smaller wake-up delay; however, the leakage power saving is less than power gating.

Figure 1.5 demonstrates the effect of supply voltage scaling on leakage power, where the x-axis represents the supply voltage and the y-axis represents the leakage power in log-scale. The result covers 3 process corners: the slow(NMOS)-slow(PMOS) corner is represented using cross markers, the fast-fast corner is represented using diamond markers and the typical-typical corner is represented using square markers. The power is measured during idle mode where no switching activities are present, and is measured for a range of supply voltages ranging from 0.5V to 1.1V. It can be seen that the leakage power reduces exponentially with the reduction in supply voltage, and 15% reduction of supply voltage results in 50% reduction in leakage power.

Leakage power reduction techniques are summarised in Figure 1.6. The left column shows active circuit leakage reduction techniques, where gate leakage can be reduced by using high-k metal gate transistors, and sub-threshold leakage can be addressed by using dual threshold standard cell design. The right column shows the idle circuit leakage reduction techniques: turning off the power supply, power gating and voltage scaling. Turning off the power supply has the lowest leakage but longer wake-up latency, while voltage scaling has the highest leakage and shortest wake-up latency.

1.3 Reliability of Digital Designs

There are different types of faults affecting the reliability of integrated circuits that can be classified into two categories: permanent faults which have lasting effects on the system reliability and transient errors which just affect the system reliability temporarily.

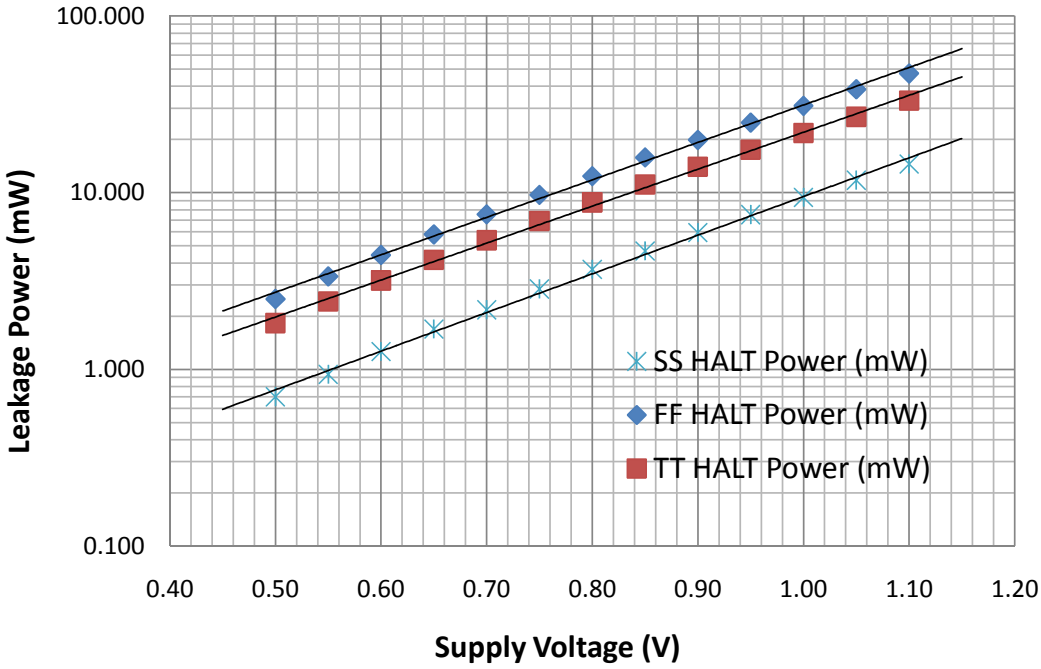


Figure 1.5: Relation between leakage power and supply voltage when ARM926 processor is in idle mode for 65nm technology

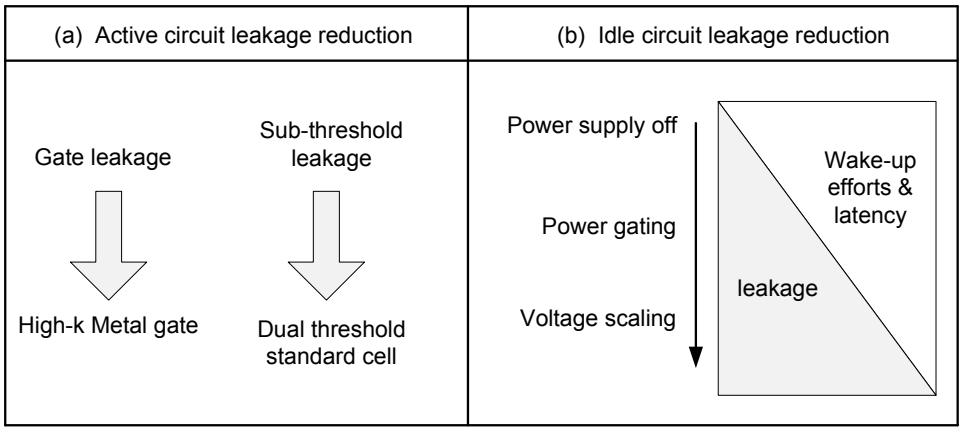


Figure 1.6: Available leakage reduction techniques and their trade-offs

1.3.1 Permanent Faults

All permanent faults can be either related to a manufacturing defect where the devices are damaged during the manufacturing process, or related to ageing effects where the devices degrade gradually during their lifetime. The fabrication of integrated circuit is very complex. Imperfect process manufacturing such as foreign particles or impurities in the silicon wafer may cause bridged connections or missing features [43, 44]. In modern ICs, the most common defects are bridges "shorts" where unintended connections form between circuit nodes, and "opens" where intended connections between circuit nodes are missing. For resistive bridge and open defects, the unintended connections are

not fully short-circuit, and for resistive-opens the broken connection is not fully open-circuit. Fault models have been developed to emulate how defects affect the operation of a circuit: the popular fault models are gate level "stuck-at" fault model [45], transistor level "stuck-at" fault model [46], bridging fault models [47] and delay fault models [48]. Automatic test pattern generation (ATPG) [44, 46] algorithms are used to generate the test patterns, and the fault coverage statistics are collected by using fault simulation. Scan chains can be used to shift the test pattern into the registers of a digital circuit. Internal scan design is the most popular design-for-test (DFT) technique which can achieve high fault coverage by isolating the combinational logic block for a complex sequential design [46].

An IC degrades gradually during its lifetime due to the various stresses it experiences in normal operations. Technology scaling reduces the feature sizes but increases the power densities and accelerates the wear-out based failures [10]. Various wear-out effects have been reported: Negative bias temperature instability (NBTI) is due to the positive charge created at the SiO2/Si interface and in the oxide under negative bias stressing, thereby causing the threshold voltage to shift and decrease the channel mobility which leads to an increase in transistor delay [49]. Gate oxide breakdown is caused by the traps formed in the gate oxide under the influence of gate-to-channel electric field. The traps accumulates and forms a resistive conduction path from gate to channel and the phenomenon is soft oxide breakdown (SBD). Once the conductive path is formed, more traps appear at an increasing pace and finally lead to hard oxide breakdown (HBD), which will increase gate leakage, delay and energy, and lower critical charge [12, 50]. Electro-migration is caused by the momentum transfer from collisions between conducting electrons and diffusing metal atoms leading to biased transport of metal atoms, which creates voids in the metal wires. It results in increased resistance which creates resistive shorts or opens in wires [51, 52]. Stress migration is due to the difference in thermal expansion coefficient between metal, silicon and oxide. When cooled from high temperatures, metal shrinks more than the surrounding medium. Therefore metal is left in a state of very high tensile stress. Coupled with electro-migration, it accelerates void creation [53]. Thermal cycling is the name given to the cyclic thermal strains due to power cycling, which can lead to the thermal fatigue failure in packaging such as short-circuit in the bonding and solder joints [54, 55].

1.3.2 Transient Errors

Transient errors are caused by electrical noise or external radiation rather than design or manufacturing defects [56]. The effects of transient errors are temporary. Inductive effects have long been known to cause problems at various levels of the system such as chip package and board level. In modern IC technology generation the inductive noise becomes an increasing concern due to the faster switching speed and higher levels of

integration [57]. Inductive noise is generated from the switching current in neighbouring circuits, it starts to cause problems in the 0.25μm technology node [58], and the problem has increased for more advanced technology nodes [59, 60]. Inductive noise also occurs in the power supply lines due to simultaneous switching noise generated by core logic and input/output circuitry, that can also impact the performance and data integrity of the electronic circuit [61].

The other source of transient errors is "soft errors" or Single Event Upset (SEU) induced by particles strikes. Most modern digital systems use synchronised designs consisting of sequential logics and combinational logics. Sequential logics are used to store the current state of the system while combinational logics are used to evaluate the next state of the system in each clock cycle. Due to the fundamental difference in their functionality the radiation event will have different effects on them.

When an ionised particle strikes a combinational logic node it generates a current pulse. The current pulse may propagate through the combinational logic tree and finally be captured by the sequential logic, which leads to state corruption. However the current pulse may not always be captured by sequential logic. There are three types of masking effects [56]: firstly, if the current pulse is small it will be attenuated by the logic tree and the pulse seen in the output will be too small to be captured by sequential logic; this is called electrical masking. Secondly, when a particle strikes a combinational logic in the early phase of the clock cycle, by the time sequential logic is clocked and the next state recorded at the output of the combinational logic tree, the current pulse generated may already faded away; this is called latching window masking. Lastly some logic gates can also mask the current pulse at one of its inputs because the output is determined by its other input values; for example if one input of an NAND gate is connected to logic '0' then the output of the gate will still be logic '1' despite the particle strike affecting the other input. In this case the particle strike has no effect on the output; this is called logic masking.

The sequential logic consists of many storage nodes; each storage node is comprised of a feedback circuit for holding the last state. When a storage node is struck by an ionised particle a current pulse is generated. If this current pulse overcomes the feedback current and exceeds the stored charge of the node, the state of this storage node is flipped so the data is corrupted. The sensitivity of a transistor to radiation-induced soft errors is dependant on its node capacitance and the supply voltage [13], both of which decrease with technology scaling. The reduction of supply voltage reduces the noise margin; the noise in the power supply rails may also impact device reliability. The failure rate of a system is the accumulation of each component's failure rate, and the increasing scale of system integration exacerbates the system reliability problem [11]. In addition, power reduction techniques such as dynamic voltage frequency scaling and power gating [37] also tend to impact the system reliability.

In this work the state integrity of digital circuit storage elements such as flip-flops or memory cells is studied. State integrity is referred as the capability of retaining the storage elements' original logic value.

1.4 Error Control Coding

Error Control Coding (ECC) is concerned with methods for delivering information from a source to a destination with minimum errors [62]. ECC provides forms of protection by introducing redundancy, in theory error rate can be reduced to a desired level with enough redundancy. ECC is also widely used for the protection of data storage elements in processors. High performance coding schemes have been developed to protect noisy communication channels; these coding schemes may need complex encoding and decoding which is not suitable for protecting an embedded processor. Hamming code and CRC code are used in this work (Chapter 3) due to the simplicity of their encoder and decoder which means smaller area, power and latency overheads. Hamming code and CRC code are both systematic codes where the original message is part of the codeword. In this section, all addition, subtraction, multiplication and division operations are GF(2) arithmetic or modulo 2 arithmetic. In GF(2) arithmetic, addition and subtraction are identical which can be implemented using an XOR gate. The implementation of multiplication uses an AND gate.

1.4.1 Hamming Code

Hamming code is a single error correcting and double error detecting code which is capable of correcting one error and detecting two errors in a codeword. For any positive integer $m >= 3$ there exists a Hamming code with the following parameters [63]:

- Code length: $n = 2^m - 1$
- Information bits: $k = 2^m - m - 1$
- Check bits: $n - k = m$

Hamming code is commonly expressed as Hamming (n, k). The encoding and decoding can be represented by generator matrix and parity check matrix. The parity check matrix H is arranged as:

$$H = [I_m \;\; Q] \tag{1.5}$$

where I_m is an identity matrix of size m. Q consist of k columns; each column is size m with Hamming weight 2 or more. For example, the parity check matrix for Hamming(7,4) is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{1.6}$$

The columns of Q can be arranged in any order without affecting the property of the code. The parity generation matrix G is arranged as:

$$G = [Q^T I_k] \tag{1.7}$$

where Q^T is the transpose of Q and I_k is an identity matrix of size k. The process of encoding can be represented by:

$$v = u \cdot G \tag{1.8}$$

where u is the information vector, v is the generated codeword vector, and G is the generation matrix. The process of decoding can be represented by:

$$s = (v + e) \cdot H^T \tag{1.9}$$

where s is the syndrome vector which indicates the error location, e is the error added to the original codeword v, and H is the parity check matrix. If there is no error, the syndrome vector s will be all zeros.

1.4.2 CRC Code

Cyclic Redundancy Code (CRC) is a linear block code with high error detection capability and low redundancy. If the information vector is interpreted as polynomial $m(x)$, the encoding can be explained as:

$$r(x) = m(x) \; mod \; g(x) \tag{1.10}$$

where $g(x)$ is the generator polynomial, and $r(x)$ is the CRC check bits which is the remainder of $m(x)$ divided by $g(x)$. The decoding is the same as encoding where another set of CRC check bits $r'(x)$ is generated. Both sets of CRC check bits are compared;

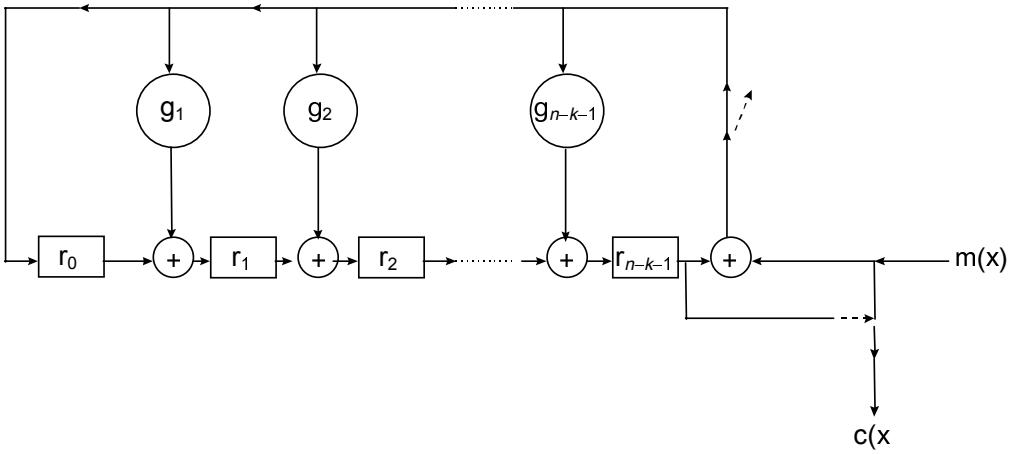


Figure 1.7: CRC generation using Linear Feedback Shift Register (LFSR) [64].

if $r(x) \neq r'(x)$, then an error is detected. The implementation can be done using an Linear Feedback Shift Register LFSR as shown in Figure 1.7. At every clock cycle, the input data is shifted into the registers; when all input bits have been processed the data remain in the registers are the CRC check bits.

1.5 Low Power Reliable Design Challenges

Figure 1.8 shows how CMOS technology scaling affects the performance, power and reliability of integrated circuits. The solid arrows represent the positive influences and the hollow arrows represent the negative influences. As can be seen, three factors are scaled with each technology generation: transistor size, supply voltage and threshold voltage. Transistor size scaling increases the performance and reduces dynamic power; at the same time it increases the static power consumption and power density and reduces the reliability. Supply voltage scaling is used to further reduce the dynamic power; however it decreases the performance and reliability. Threshold voltage reduction is used to improve the device performance but increases the static power at the same time. The power density of the chip increases with the reduction of transistor size, and it also increases with device dynamic and static power. The increase of the power density leads to higher temperature, which will further increase the power consumption [38]. To summarise, the transistor performance and dynamic power consumption benefit from technology scaling while static power consumption and system reliability are affected negatively. These technology scaling associated problems such as increase in leakage power, worsened process variation, and resulting in degradation in reliability and increasing levels of complexity need to be addressed. Power reduction techniques have been extensively used to address the increasing power density problem; however they have detrimental effects on reliability. Therefore maintaining reliability while reducing power consumption with minimum cost is the aim of this research.

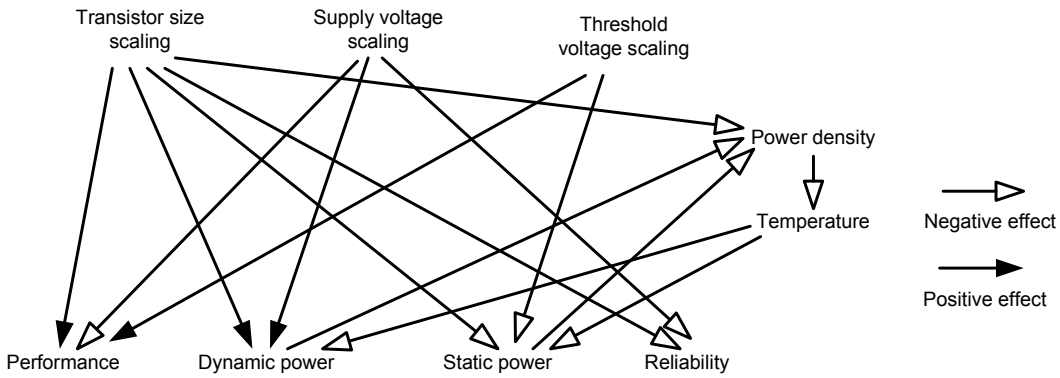


Figure 1.8: Technology scaling and its effects on performance, power and reliability

1.6 Thesis Organisation

This thesis presents three complimentary but orthogonal contributions. Chapter 3 presents the first contribution that deals with improving reliability of states stored in state retention registers, which are affected by fluctuations in the supply rail due to rush current in the power supply network when power gating logic is turned on from sleep mode. It is shown that this problem is further exacerbated because the critical charge of a storage node is affected by the combined effect of process variation and supply voltage scaling. To improve state protection, a combination of hardware and software based error monitoring and recovery method is proposed for single and multi-bit error recovery. Through two case studies, including implementation of a commercial embedded processor (ARM Cortex^{TM}-M0), it is shown that it is possible to achieve more than 2 orders of magnitude improvement in system reliability compared to a design without error protection, when considering maximum soft error rate observed at normal operating conditions. Area overhead is minimised through reuse of the scan chains for state encoding using ECC codes.

Chapter 4 presents the second contribution that uses measured results from 82 test chips to characterise state integrity challenges of voltage scaled flip-flops in the presence of process, voltage and temperature (PVT) variations. Measured results show that the minimum retention voltage of a design varies across dies and therefore a single retention voltage for all dies lead to overall higher leakage power consumption during sleep mode. A novel solution is presented to address this state integrity challenge, while minimizing leakage power of flip-flops during sleep mode. The proposed solution uses a characterisation algorithm to minimise data retention voltage for each die thereby reducing leakage power and uses a control flow for error detection and single bit error recovery. Silicon results show that at characterised data retention voltage, state integrity is preserved, while achieving up to 17.6% reduction in retention voltage across the 82 dies, thereby saving leakage power.

Chapter 5 presents the last contribution that deals with the reliability of the complete memory system and describes an architectural simulation-based framework for joint optimisation of reliability, energy consumption and performance. Memory systems are highly susceptible to soft errors and blanket protection of every part of memory system through ECC is not cost effective. Therefore, an accurate estimation of memory reliability with targeted protection is proposed. The proposed solution identifies and protects the most vulnerable part of the memory system to minimise protection cost. Furthermore, local level-1, L1 resizing together with voltage and frequency scaling is proposed for further energy savings while maintaining performance and reliability of an embedded processor system. Detailed evaluation shows that energy reduction is possible for most applications in a reliability constrained design.

These three contributions presented in this thesis are supported by detailed analysis using state-of-the-art design automation tools, and in-house software tools and, where necessary, are validated through FPGA implementation of commercial low power embedded processors and measured results from fabricated designs.

1.7 Publications

The contributions of the research presented in this thesis have been published as follows:

1.7.1 Journal Publications

1. Sheng Yang, Saqib Khursheed, Bashir M. Al-Hashimi, David Flynn and Sachin Idgunji, "Reliable State Retention-Based Embedded Processors Through Monitoring and Recovery", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol.30, no.12, pp.1773-1785, Dec. 2011
2. Sheng Yang, Saqib Khursheed, Bashir M. Al-Hashimi, David Flynn and Geoff V. Merrett, "Improved State Integrity of Flip-Flops for Voltage Scaled Retention under PVT Variation", Circuit and Systems-I, IEEE Transaction on (In press)
3. Sheng Yang, Saqib Khursheed, Bashir M. Al-Hashimi, David Flynn, "Modeling Framework to Optimise Memory System Reliability", ACM Transactions on Embedded Computing Systems (Under preparation)

1.7.2 Conference Publication

1. Sheng Yang, Saqib Khursheed, Bashir M. Al-Hashimi and David Flynn, "Scan Based Methodology for Reliable State Retention Power Gating Designs", Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.69-74, 8-12 March 2010

Chapter 2

Literature Review

This chapter provides a broad overview of the state-of-the-art research that is related to this thesis research topic. To complement this literature review, each of Chapters 3, 4 and 5 has its own literature review, placing the contributions described in these chapters in the context of the relevant research. Section 2.1 reviews the unreliability source of power-gated designs and discusses the principles of some of the key reported techniques in addressing this reliability issue. Section 2.2 reviews the reliability in supply voltage scaled designs and outlines briefly the main techniques reported in improving the reliability of such designs. Analysing the reliability of processors requires the availability of metrics that quantify their susceptibility to failure. Recent research efforts indicate that process, voltage and temperature (PVT) variation is affecting the reliability of digital designs, Section 2.3 discusses briefly why this is the case and describes some of the reported techniques in mitigating theses variation. Part of transient errors referred to as soft error is induced by particle strike, Section 2.4 describes the source of soft errors, its impact on digital circuit reliability and existing mitigation techniques. Section 2.5 reviews the metrics proposed and some of the architectural innovations reported to improve the reliability of embedded processors. Finally, this chapter includes the objectives for the research described in subsequent chapters in light of the reviewed research.

2.1 Reliability of Power-gated Designs

In Chapter 1 Section 1.2.2 various leakage reduction techniques were described. Power gating is effective in reducing leakage power, but switching on/off the power supply of a logic block can cause voltage fluctuation in the supply and ground rail; this phenomenon is also called ground bounce. The power supply rails of an integrated circuit are not ideal, as wires have resistance and there are capacitance and inductance between the wires. Figure 2.1 shows a simplified model of power supply rails in a power-gated design.

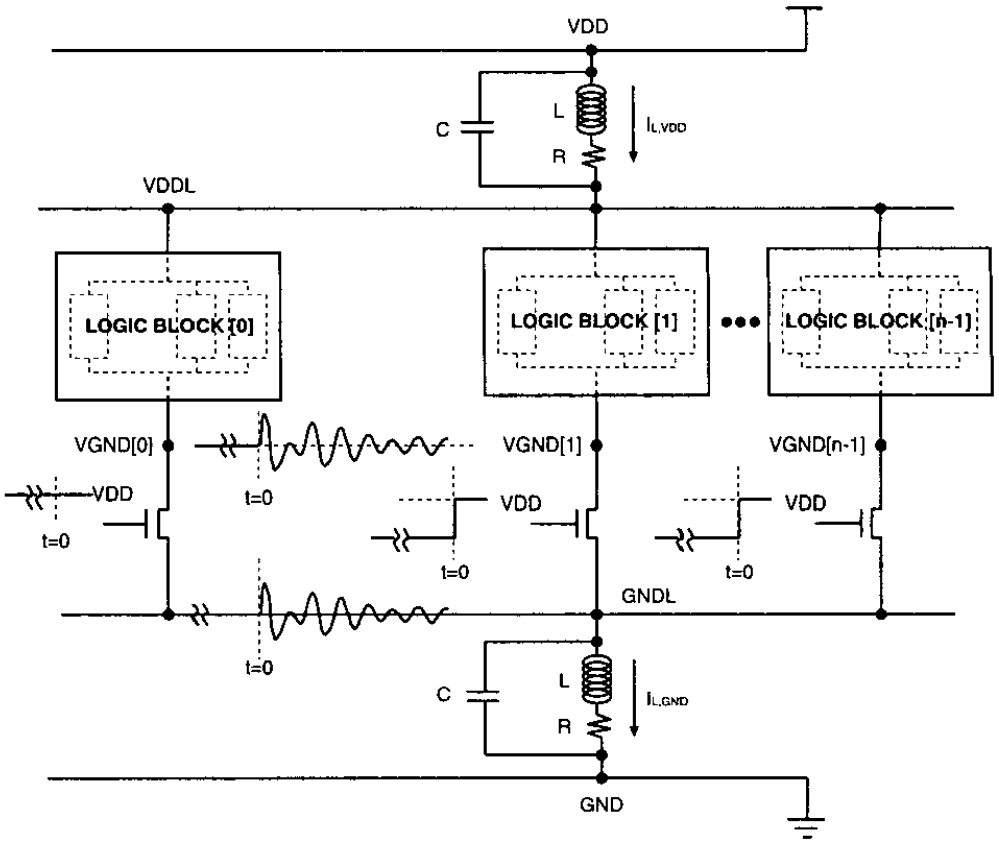
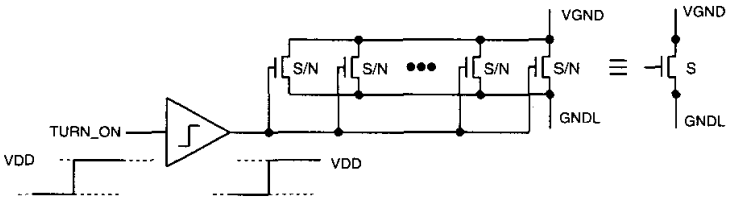


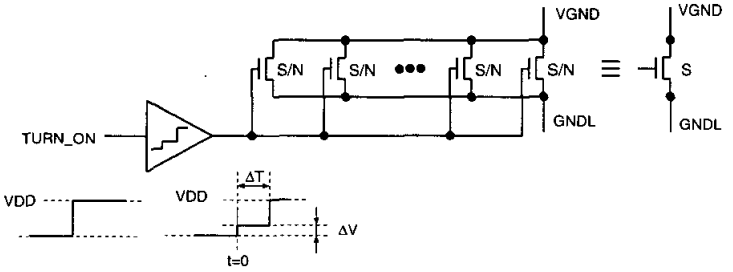
Figure 2.1: Ground bounce in power-gated design [65].

In sleep mode, when the power gating transistors (power switches) are off, the internal capacitance of the power-gated logic blocks are discharged near ground level and the leakage currents is limited by the power switches. When the power-gated logic blocks are reactivated, the power switches are turned on, the internal capacitance of these logic blocks are rapidly charged up. The rapid charge of the logic block causes a large in-rush current, and the sudden change of current induces a voltage across the wires' inductance, which can be modelled as step response of an RLC circuit [65, 66]. The voltage fluctuation at the power supply rails may corrupt the state retention elements that hold the circuit states during sleep mode, which leads to a potential reliability problem.

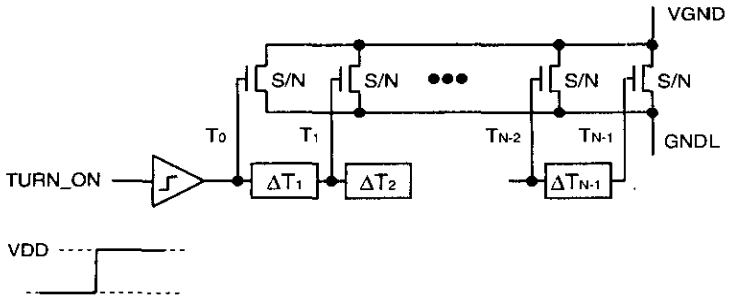
Several techniques were used to reduce the amount of ground bounce cause by power switches [65]. Figure 2.2.(a) is the conventional power switches arrangement, where a control signal *TURN_ON* is connected to an array of power switches. In this arrangement all power switches are turned on at the same time, causing a large in-rush current. To reduce this in-rush current, Figure 2.2.(b) shows the voltage level of control signal *TURN_ON* increases over time in a step wise manner. Figure 2.2.(c) shows that the control signals of power switches are connected in a buffer chain; the power switch at the beginning of the buffer chain is turned on first and the subsequent switches are



(a) Conventional power switches arrangement



(b) Power switches with step wise control



(c) Power switches with delayed control signals

Figure 2.2: Change power switches arrangement for ground bounce reduction [65].

then turned on one by one. The interval is controlled by the delay of the buffers. This method has been incorporated by foundries into standard cell libraries where the power switches' control can be connected in a daisy chain. Pump capacitors was proposed to slowly turn on the power switches and use a voltage monitor circuit to detect the end of the activation process [67]. Other techniques are stacking power switches [68], raising voltage level of virtual ground and virtual supply [69, 70], and controlling the input vector of logic block [71]. These works protect the power-gated design by reducing the ground bounce through in-rush current reduction.

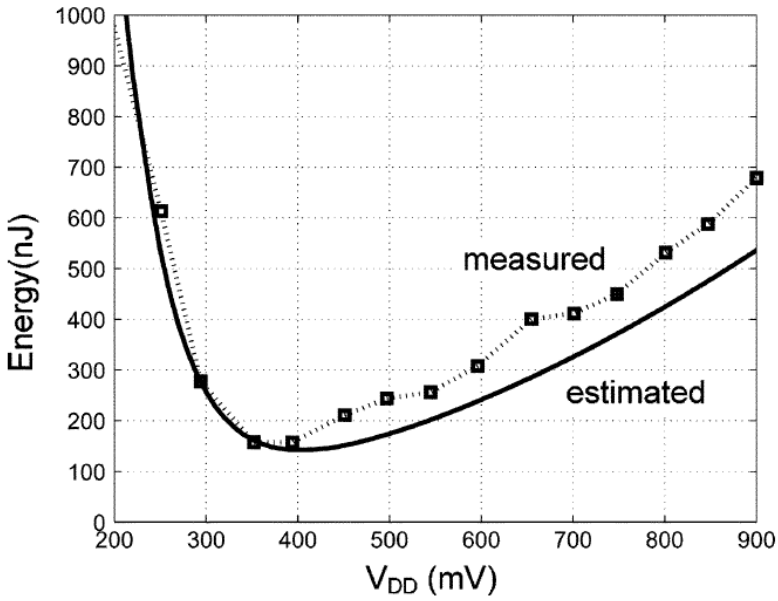


Figure 2.3: Energy dissipation under different supply voltage [74].

2.2 Reliability of Voltage Scaled Designs

As discussed in Chapter 1 Section 1.2, supply voltage scaling is an effective power reduction technique for both dynamic power and leakage power. Recent studies have seen dynamic power reduction of 20% [72] and 32% [73] using DVFS. For low performance applications requiring energy minimization, a CMOS circuit can be designed to operate in the sub-threshold region [74]. Figure 2.3 shows the energy dissipation under voltage and frequency scaling. It can be seen that the minimum energy point is under 350-mV supply voltage, at which point the energy efficiency is 4x better compared to 0.9-V supply voltage. However, to operate under sub-threshold voltage, specially designed gate libraries are required [74]. Sub-threshold operation exacerbates the impact of process variations on gate delay which is as high as 300% [75].

Supply voltage scaling is also an effective technique for leakage power reduction. Power minimisation through supply voltage scaling during sleep state has been proposed in [41, 76], where voltage scaling is implemented by using IR-drop of diode to reduce power supply. Recent studies have shown that voltage scaling effectively reduces sub-threshold and gate-leakage power in Deep Sub-Micron (DSM) designs [38, 77]. This is because of the negative exponential relationship of leakage power and supply voltage (Chapter 1, Section 1.1, Equation 1.4). Unlike sub-threshold designs, when voltage scaling is used for leakage reduction, the most energy efficient point is the minimum supply voltage required to hold the data in storage elements of a circuit. Data Retention Voltage (DRV) is the voltage point at which a SRAM cell can still hold its original value [78, 79]. A recent study using Monte-Carlo simulations showed that within-die process variation causes the distribution of DRV for 90nm and 45nm SRAM cells [78]; and the studies on cache showed the same trend [80–82]. For correct operation the supply voltage should be

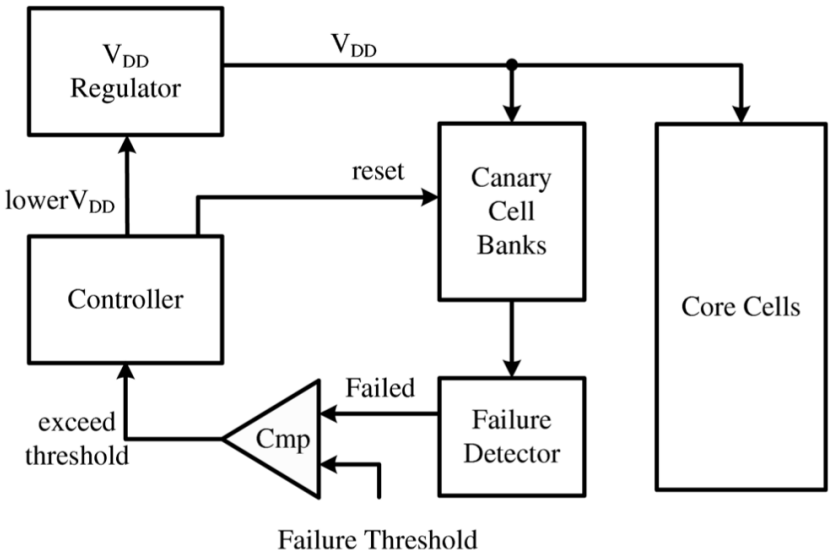


Figure 2.4: Archiecture of the Canary-based feedback loop for SRAM standby V_{DD} scaling [79].

kept above the highest DRV of all SRAM cells, which is 150-mV for 90-nm and 320-mV for 45-nm (Figure **??**).

Canary cells were proposed to provide a safety margin for voltage scaling [78, 79, 83, 84]. Canary cells are an array of specially designed storage cells which fail at higher supply voltage than the protected design. Figure 2.4 shows the architecture of a canary-based design. The voltage regulator is used to control the supply voltage of the core cells which can be lowered to reduce leakage power. Banks of canary cells are designed to fail across a range of voltages which is above the highest DRV of the core cells. The failure of Canary cells is monitored by the failure detector which is compared to the failure threshold. If the failure voltage exceeds the pre-set margin, the controller raises the supply voltage to the last working value. Figure 2.5 shows the schematic of Canary cells, which must fail before the core cells to prevent the loss of data; so a PMOS transistor M8 is inserted between the supply rail and Canary cell to provide additional IR drop, the resistance of M8 can be controlled by signal VCTRL. Canary design provides 11x leakage power saving compared to nominal supply voltage. Other techniques using supply voltage scaling for leakage reduction are post silicon characterisation method for fine-grain cache line supply voltage control [80] and for SRAM retention voltage adjustment in individual die [81, 82].

2.3 Impacts of Process Variation on Reliability

In DSM technology, the gate oxide is only a few atoms thick, channels contain countable dopant atoms [85], and the size of transistor is an order of magnitude smaller than the wavelength of the light used in lithography to create these devices [86]. The uneven

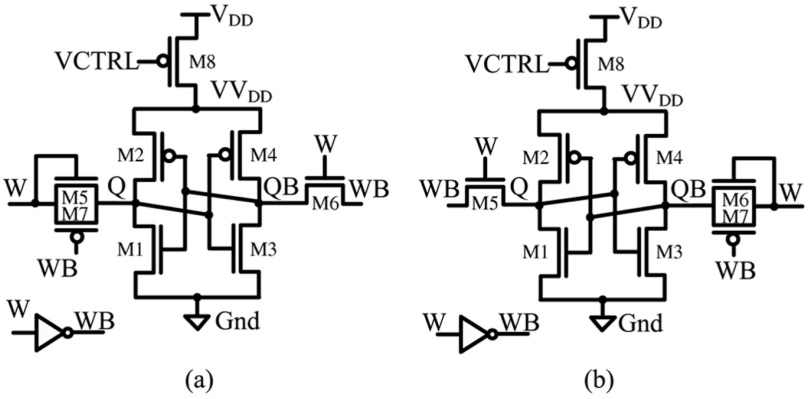


Figure 2.5: Schematic of Canary cell (a) for storing '1' and (b) for storing '0' [79].

distribution of dopant atoms or the uneven surface of gate oxide will cause a significant difference in the electrical property of a transistor. The difference between geometries drawn in CAD systems and the implemented devices will cause uncertainty in terms of performance, power and reliability. This uncertainty will continue to grow as the scaling trend continues [30].

The main causes of process variation are random dopant distribution, sub-wavelength lithography and line edge roughness [87]. The result is the variation in the transistor's oxide thickness, carrier mobility, channel length and threshold voltage [88–91]. The number of dopants in the depletion region of a DSM technology is in the order of hundreds; therefore the microscopic dopant distribution in the MOSFET channel will influence the device electrical performance [92]. The effect of random dopant distribution on threshold voltage can be tens of millivolts when the supply voltage and threshold voltage of DSM are about 1.2-V and 400-mV respectively [93]. Lithography is the process of printing circuits by shining light through photomask to the surface of silicon wafer coated with a photosensitive material (resist); therefore it transfers the mask pattern into the wafer [86]. The main problem in sub-wavelength lithography is diffraction and interference, which makes lines printed on the wafer shorter or wider than those on the mask [94]. Line edge roughness is roughness along the transistor channel caused by number of statistically fluctuating effects in the lithography and chemical etching processes such as photon flux variations, distribution of chemical species in the resist, random walk nature of acid diffusion and resist polymers being dissolved during development [95]. Line edge roughness reduces transistor channel length by about 5-nm when technology scaled below 100-nm [96], which can lead to threshold variation of up to 20-mV [97, 98]. From measurement results of 65-nm test chip, a recent study has shown that threshold voltage (V_{th}), effective gate length (L_{eff}) and carrier mobility (μ) are the main parameters affecting the performance of CMOS transistors. The results are shown in Figure 2.6; the variation of V_{th}, L_{eff} and μ follows Gaussian distribution with standard distribution (σ) of 5%, 4% and 21% respectively.

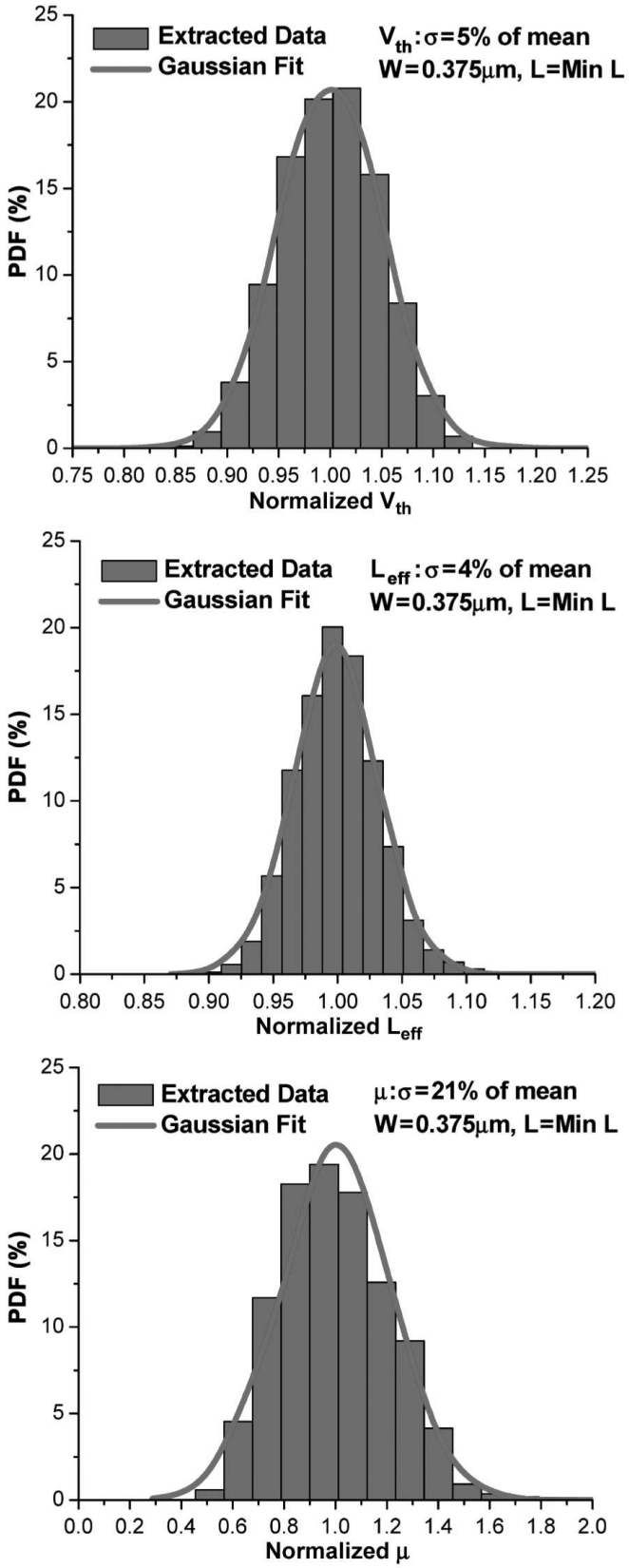


Figure 2.6: Variations of threshold voltage, effective channel length and carrier mobility for 65nm CMOS [90].

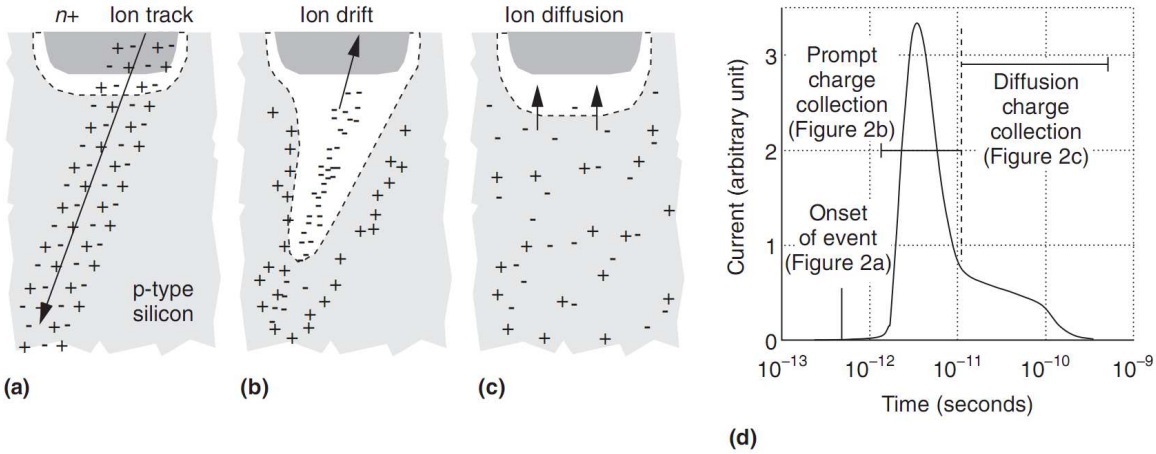


Figure 2.7: Charge collection in a silicon junction (a) after an ion strike, (b) during drift collection, (c) during diffusion collection, and (d) junction current induced as function of time [13].

2.4 Impacts of Soft Errors on Reliability

Ionised particles can cause temporary data corruption in digital ICs [13]. Ionised particle strike leaves a track of electron-hole pairs with high carrier concentration as shown in Figure 2.7.(a). After the strike the electric field in the depletion region quickly (within tens of picoseconds) collects carriers creating large drift current at the node as shown in Figure 2.7.(b). Lastly, the remaining carriers will diffuse into the depletion region over a longer time period (hundreds of nanoseconds) as shown in Figure 2.7.(c). Figure 2.7.(d) shows the junction current pulse as the result of the particle strike as the function of time, and the x-axis is in log-scale. The short current pulse with large magnitude is created by the drift current and the following long current pulse is the result of the carrier diffusion. There are three radiation sources [13, 99] that produce ionised particles: Alpha particles emitted by trace uranium and thorium impurities in packaging materials; high-energy neutrons from cosmic radiation which react with silicon nuclei producing ionised particle, and low-energy neutrons from cosmic radiation which interact with isotope boron-10 in IC materials, specifically in borophosphosilicate glass (BPSG). Alpha particle radiation has been reduced by using high purity materials and by physically separating and shielding the sensitive circuit components from the alpha particle emitting material. The BPSG insulator layers have been replaced with dielectric material free of boron-10, which eliminates the problem caused by low-energy neutrons. The ionised particle produced by the high-energy neutron is the major contributor to soft errors in modern digital IC technology. A recent study has shown that soft error can be 1000 times more likely to happen than hard failures in modern electronics systems [13]. The reduction of transistor size and the ever-increasing integration scale exacerbate the problem.

When a storage node is struck by an ionised particle, a current pulse is generated; if this current pulse overcomes the feedback current and exceeds the stored charge of the node, the state of this storage node is flipped so the data is corrupted. The amount of the charge required by the current pulse to flip a storage node is known as its critical charge [100]. Therefore the reliability of sequential logics can be obtained through the study of its critical charge. This transient error that is induced by particle strike is also call soft error. For an individual device, Soft Error Rate (SER) is usually represented by λ with the unit of $failures/s \cdot bit$. A method for estimating Soft Error Rate (SER) in CMOS SRAM cell was proposed [101]:

$$\lambda \propto F \times A \times exp(-\frac{Q_{crit}}{Q_s}) \tag{2.1}$$

where F is the high energy ($> 1MeV$) neutron flux, A is the area of the circuit sensitive to particle strikes, Q_s is the charge collection efficiency of the device, and critical charge (Q_{crit}) is the amount of charge required to change the original value in a storage element resulting a transient error called soft error or single event upset (SEU) [102]. Therefore the critical charge of a storage node determines its sensitivity to environmental noises. In recent studies [103, 104], critical charge has been modelled as:

$$Q_{crit} = C_N \cdot V_{DD} + I_{DP} \cdot T_F \tag{2.2}$$

where C_N is the storage node capacitance, V_{DD} is the supply voltage, I_{DP} is the maximum current of the on PMOS transistor and T_F is the cell flipping time. Clearly, critical charge of a storage node reduces linearly with supply voltage and node capacitance. Critical charge also reduces with the advance in technology scaling; a flip-flop designed using 45-nm technology showed lower critical charge than the one designed with 65-nm technology when comparing them at their respective nominal operating voltages [12]. In Section 2.3, it was shown that process variation has significant impacts on Deep Sub-Micron (DSM) designs. The effect of supply voltage scaling and process variation on critical charge of a flip-flop was investigated using 40-nm CMOS technology. The results are shown in Figure 2.8. Two observations can be made: firstly, the critical charge of a flip-flop reduces with the reduction in supply voltage; and secondly, the graph also shows the worst-case critical charge of a flip-flop reduces further under 1σ, 2σ and 3σ process variations. These results demonstrate that the critical charge of a flip-flop is not only reduced by supply voltage reduction but also by process variation. The reduction in critical charge can lead to the increase in soft error rate [105], therefore affecting the reliability of digital designs.

Soft error mitigation techniques for storage elements can be classified into circuit level hardening, hardware redundancy and time redundancy [11]. Circuit level hardening involves adding resistance [106], capacitance [107] or feedback loop [108] to the storage

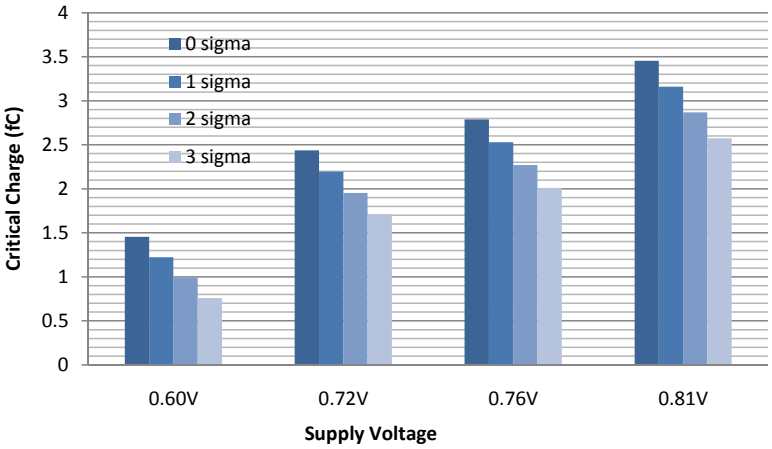


Figure 2.8: Critical charge and supply voltage in the presence of process variation.

node or combining the above techniques [109]. Hardware redundancy either duplicates the storage elements [11] or uses error control coding [110]. Time redundancy is achieved through re-execution of software programs [111, 112].

2.5 Embedded Processor Reliability

There are two ways to analyse the reliability of a system: Statistical fault-injection injects faults into random part of the system and analyses the output of the system [113–116], and a Architectural vulnerability analysis calculates the probability that a fault in the system will result in an erroneous output [117–119]. Fault-injection models the behaviour of faults and emulates the problems a system will experience when faults occur. It is a useful tool to study how a system behaves under faulty conditions and to evaluate its fault tolerance capability. The limitation of faults injection is very long simulation time if the fault rate is low, which is the case for most systems under normal operating conditions. To address this problem, Architectural Vulnerability Factors (AVF) is proposed [117] to indicate the probability that an bit-flip of a storage cell will result in an visible external error. The system error rate can be estimated by the product of its raw fault rate and its AVF. There are some limitations: firstly, to undertake AVF analysis of a underlining system, knowledge of its architecture is required; secondly, it can estimate the probability of erroneous output but cannot predict the impacts of these faults. Therefore fault-injection and AVF analysis are two complementary techniques in system reliability analysis.

In a processor system, storage elements hold circuit states and program data. Micro-architectural registers, register files and memory system including caches and main memory are some examples. Logic faults in storage elements may lead to unpredictable system behaviour. Using AVF analysis, the reliability of processor core was investigated

in a number of studies [117–120]; and the reliability of caches was examined in other studies [121–124]. To protect register files, inherent redundant resource of a processor is commonly utilised. In [125], the values in active used registers are duplicated in the unused registers; whereas in another study [126], when 64-bit registers are used to store 32-bit values, the 32-bit values are duplicated and stored in both halves of the 64-bit registers. Two redundancy techniques are combined in a further study [127]. In modern processors, multi-threading and out-of-order execution are commonly used. Multi-threading is used to improve processor reliability by duplicating process and thread execution [128–132]. Multiple execution units in an out-of-order processor are used for instruction level redundant execution [133–137]. In cache memory, parity caching was proposed to protect the most used cache lines [138]. Different coding schemes are used to protect clean and dirty cache lines [139]. Two dimensional error coding was proposed to provide multi-bit error tolerance in cache memory [140]. Inactive cache block is used to provide redundancy for active cache block [141].

2.6 Research Objectives

There is little doubt from this literature review that performance, power and reliability are tightly coupled particularly for Deep Sub-Micron (DSM) technology nodes. For DSM technology nodes, power density is a constricting factor limiting the performance and future scaling. In the past decade, digital circuit power reduction has attracted a large amount of research efforts targeting both dynamic power and leakage power. For dynamic power reduction, reducing switching activity and supply voltage scaling are the main focus. For leakage power reduction, tuning transistor threshold voltage, cutting off power supply and supply voltage scaling are popular tools. The manipulation of supply voltage is very effective in power reduction; at the same time it affects the state integrity of storage elements. Process variation has been kept under sufficient control for 40 years until silicon manufacturing was scaled down to DSM such as 65nm node [85]. Process variation is also affecting the reliability of digital circuit due to the mismatching of electrical property between the final product and the intended design, and the difference grows with the reduction in transistor size. This means that, in future technology generations, process variation can only get worse and the behaviour of digital circuits is becoming less predictable. The combination of process variation and power reduction techniques has detrimental effects on the reliability of digital circuits.

The main focus of this thesis is to investigate the impact of power reduction techniques on digital circuit reliability in the presence of PVT variation, and to develop low-cost and effective protection techniques to improve reliability. EDA tools are heavily used in modern digital circuit designs, because of the increased complexity which makes automation an essential component of design flow. The developed techniques should be compatible with commercially available EDA tools and should avoid the manual

insertion of protection circuitry. This is vitally important to minimise design effort which helps to reduce time to market, risk and cost associated with development and implementation. To study the effectiveness and the cost of the proposed techniques, FPGA implementation and test chip fabrication are needed.

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

- Investigate the impact of power reduction techniques on the reliability of embedded processors in the presence of soft error and PVT variation.
- Develop effective mitigation techniques to improve the reliability of low-power embedded processors registers and memory systems, including the development of modelling framework for joint optimisation of reliability, energy and performance.
- Validate the developed techniques using simulations and experiments through FPGA implementation and chip fabrication based on commercial processors.
- Incorporate the developed techniques into industry standard EDA design flows with the aim to produce automated design flow for reliable and low-power processor designs.

Chapter 3

Improving The Reliability of State Retention Designs

In Chapter 1 Section 1.1, the impact of technology scaling was discussed such as the fact that transistor size scaling reduces node capacitance which in turn reduces critical charge making it more sensitive to external noise sources. In embedded applications, idle time power reduction is frequently exploited to minimise energy consumption [142,143], which is referred to as sleep mode in this work. In Chapter 1 Section 1.2.2, it was shown that state retention power gating [37] and supply voltage scaling [41] are two state retention design techniques for effective idle circuit leakage power reduction. During sleep mode, circuit states are stored in a retention latch structure (referred to as balloon circuit [144] in the case of power gating; circuit states are stored in flip-flops but with reduced supply voltage in the case of supply voltage scaling. Circuit states may be corrupted due to supply voltage fluctuation in the case of power gating, or critical charge reduction in the case of supply voltage scaling. A number of techniques exist to reduce voltage fluctuation of power gating design in order to reduce the impact on circuit reliability as discussed in Chapter 2, Section 2.1. However, voltage fluctuation cannot be eliminated completely and circuit states are essential for correct operations. Furthermore, some applications may have low duty cycle [145], where the processor spends most of its time in sleep mode. For these applications power-gated circuit states are more vulnerable than active circuit states because the error rate is proportional to the exposure time. Concurrent ECC protection on unstructured flip-flops such as control registers and finite state machines has high area and power overheads particulary for error correcting code [146,147]. This motivates the need to develop cost-effective techniques to ensure power-gated circuit state integrity.

The key objective of this research is to develop effective mitigation techniques to improve the reliability of embedded process registers. This chapter proposes a novel method for improving reliability of state retention registers in low power designs that is compatible

with standard EDA tools, and uses two case studies to show the effectiveness and the cost of the proposed method. The rest of the chapter is organised as follows: Section 3.1 proposes scan-based state monitoring and recovery method for state retention designs, including state monitoring and recovery architecture, inserting state monitoring control into conventional state retention flow and integrating the protection circuitry using conventional synthesis flow. Section 3.2 and Section 3.3 describe two case studies to improve the reliability of a FIFO circuit and an embedded processor (ARM Cortex^{TM}-M0) using the proposed method. Both design are validated using FPGA and synthesised using ST 65nm technology library for implementation trade-offs analysis. Section 3.4 concludes this chapter.

3.1 Proposed Method

Error detection and correction codes have been used extensively to improve the reliability of memory circuits [148]. This is achieved by generating parity bits when writing data into memory and checking the data against saved parity bits when they are read out of memory. For a given design, flip-flops are not as structured as memory blocks; they are scattered physically. These flip-flops do not have a unified input and output channel which is required to access their data and generate parity bits like memories. Scan chains [46] which connect flip-flops into long shift registers for performing manufacturing test provide the channel for parity bits generation. Scan chains insertion is normally automated by EDA tools, and involves replacing the system flip-flops with scan-enabled flip-flops and creating scan-in, scan-out ports and a scan-enable signal without affecting the functionality of the original design. Only when the scan-enable signal is active are the flip-flops reconfigured into a daisy chain and the scan-in and scan-out ports are the input and output of these chains.

The scan chains of a design are exploited for the purpose of protecting its state integrity. This is achieved by monitoring the states through scan chain encoding and decoding. The method consists of three main steps. Firstly, the state monitoring and recovery architecture is described (Section 3.1.1). Secondly, a controller is designed to generate the control signals for state monitoring and recovery (Section 3.1.2), Finally a synthesis flow is developed to incorporate the first two steps into conventional state retention designs by automatic generation of protection circuitry and control signals (Section 3.1.3).

3.1.1 Step 1: Architecture

The architecture of a circuit under protection (CUP) with state monitoring and recovery block (SMRB) is shown in Figure 3.1, which consists of a state monitoring block and a error correction block. The scan chain inputs to the CUP are controlled by a multiplexer

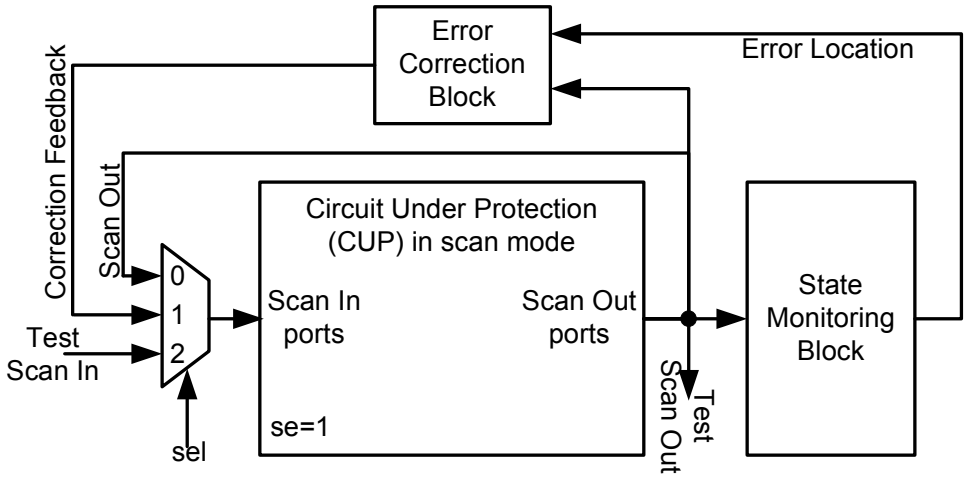


Figure 3.1: Architecture of State Monitoring and Recovery Block (SMRB).

with a 'sel' signal, which allows the scan chains to be recirculated or conditionally corrected (inverted), or used as conventional manufacturing scan inputs. The scan enable, 'se', to the CUP is asserted to configure all the registers into scan-shift mode rather than functional mode. Before switching to sleep mode, the state monitoring block encodes the CUP states, where signal 'sel' is set to '0' and 'se' is set to '1'. The CUP is put into scan mode, and its scan-out ports are connected to its scan-in ports and the state monitoring block. Assuming that each scan chain contains 'l' flip-flops, by circulating the scan chains for 'l' clock cycles, the state monitoring block generates and stores the parity bits of the CUP states. After switching to active mode, the state monitor decodes the states, where signal 'sel' is '1' and 'se' is '1'. The CUP is put into scan mode, and its scan-in ports are connected to the error correction block, its scan-out ports are connected to the state monitoring block and the error correction block. The state monitoring block checks the states of the CUP against the stored parity bits. When errors are detected, the state monitoring block sends the error locations to the error correction block which corrects the corrupted states and feeds back to the circuit. In manufacturing test mode, the control signal 'sel' is set to '2', and the scan enable signal 'se' is controlled by the tester; the CUP's scan-in ports are connected to the test scan-in ports and its scan-out ports are connected to the test scan-out ports. The proposed approach benefits from reusing the manufacturing scan chains for system state monitoring and recovery as demonstrated next.

State monitoring is done using hardware error detection code (CRC or Hamming) [63] and state recovery is done using either hardware error correction code (Hamming) or software state recovery. The method has no impact on the design's timing (critical path) in normal mode. This is because all state monitoring is done in scan mode and the scan path is not part of the critical path. However it has a cost in terms of area overhead, wake-up latency and energy consumption (details are shown in Section 3.2.3 and Section 3.3.4). In terms of power consumption, the main contributor is the scan

chain switching power, which is CPU-state pattern dependent and in the worst case can result in every flip-flop toggling in every clock cycle. The scan power needs to be analysed and scan power reduction techniques for toggle suppression [149] can also be used to reduce the state monitoring power dissipation to avoid exceeding the power distribution integrity for the CUP. For the state monitoring block to generate parity bits, the circuit's states need to be circulated through the scan chains. If the number of registers in each scan chain is 'l' and the clock period is 'T', the encoding and decoding time will be '$l \times T$'. If 'l' is large, each encoding and decoding cycle can take a long time and therefore consume a significant amount of energy.

The scan chains can be configured to reduce encoding and decoding time but also can be reconfigured for the standard manufacturing test. An example is shown in Figure 3.2, where a long serial manufacturing test scan chain can be partitioned into 4 parallel scan chains using 4 multiplexers. Signals 'SI' and 'SO' are connected to manufacturing test scan-in and scan-out port; manufacturing test is enabled by setting 'TE' to '1'. When 'TE' is set to '0', the scan chains are reused for state monitoring and recovery. In another example, assuming the test scan width (I/O width for manufacturing test) is 4-bits, and the state monitoring block employs Hamming (7,4) code to monitor circuit's states with input width of 4-bits per state monitoring block; originally, a circuit had 128 flip-flops connected in 4-scan chains, and since state monitoring block's width is 4-bits, it will take $(128 \div 4) = 32$ clock cycles for encoding and decoding the data. Next, consider that 128 flip-flops are re-ordered into 16 scan chains allowing 4 state monitoring blocks to work in parallel. The number of encoding and decoding clock cycles will then be $(128 \div 16) = 8$, resulting in 4x speed-up. The latter 16 scan chain configuration is shown in Figure 3.3 (a); as can be seen, 4 state monitoring blocks are operating in parallel taking data from 128 flip-flops configured in 16 scan chains. Figure 3.3 (b) shows how this configuration can be re-used for 4-bit scan chain operation during manufacturing test: the output of So[3:0] is fed back to Si[7:4] and so on, until test data is scanned out through So[15:12]. To reduce latency and energy consumption, shorter scan chains are needed; but at the same time, this will increase the area overhead due to additional state monitoring blocks. To make each scan chain shorter, the number of scan chains can be increased. The detailed area, energy and latency trade-offs related to scan chain configuration are discussed in Section 3.2.3.

3.1.2 Step 2: Controller

A state monitoring and recovery controller is designed to provide the controls for the scan enable signal 'se' and the input multiplexer selection signals 'sel' (Figure 3.1) for encoding where 'se' is set to '1' and 'sel' is set to '0', and for decoding where 'se' is set to '1' and 'sel' is set to '1'. The method targets state retention designs with two different operation modes: active mode and sleep mode. Active mode (normal mode of

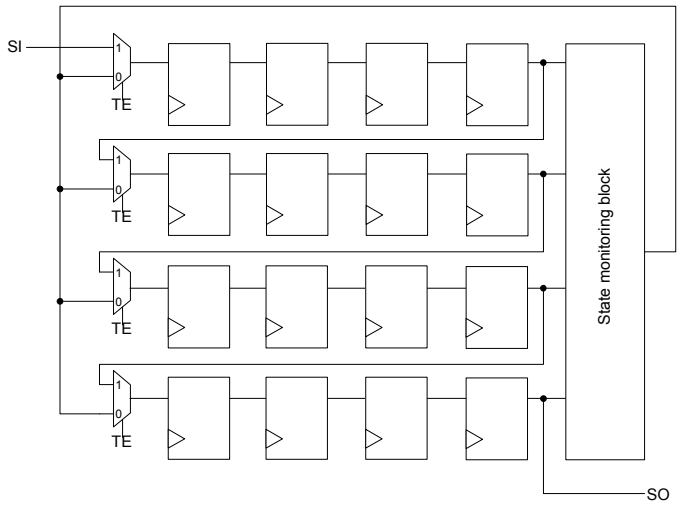


Figure 3.2: Reuse and partition manufacturing test scan chain for state monitoring.

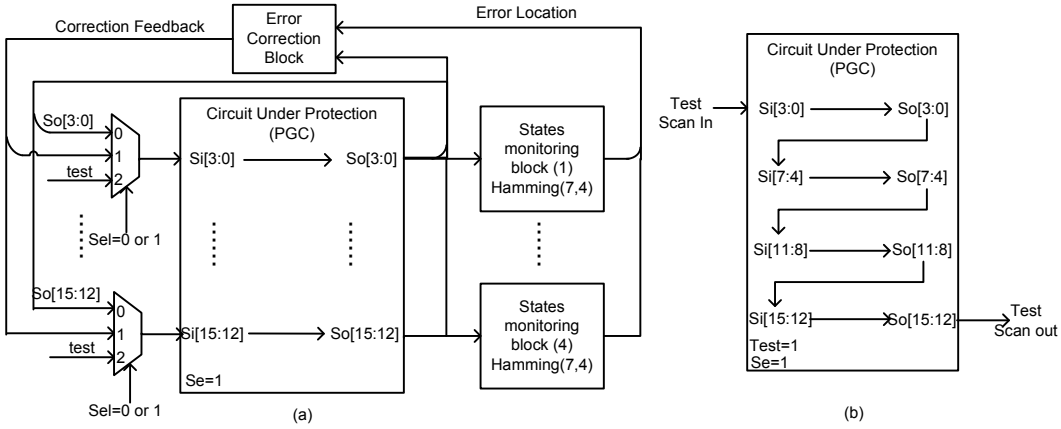


Figure 3.3: Scan chain configurations (a) state monitoring (b) manufacturing test.

operation) is the same for both power gating and supply voltage scaling. For sleep mode, in the case of power-gating, the state is saved in retention registers and power supply of the design is turned off, in the case of supply voltage scaling the supply voltage of the design is reduced aggressively to the retention voltage where the registers of the design can still hold their states. The sleep-active mode transition control is similar for both power gating and supply voltage scaling, except that power gating requires additional control for saving and restoring circuit states. In this implementation power gating is demonstrated.

The state monitoring and recovery control can be integrated into a conventional sleep-active mode transition control flow which is illustrated in Figure 3.4. Figure 3.4 (a) shows the conventional sleep control flow. Assume a circuit starts from active mode. When signal 'sleep' is '1', the sleep controller starts the sleep sequence, for power gating design it includes saving the circuit's states and turning off the sleep transistors while

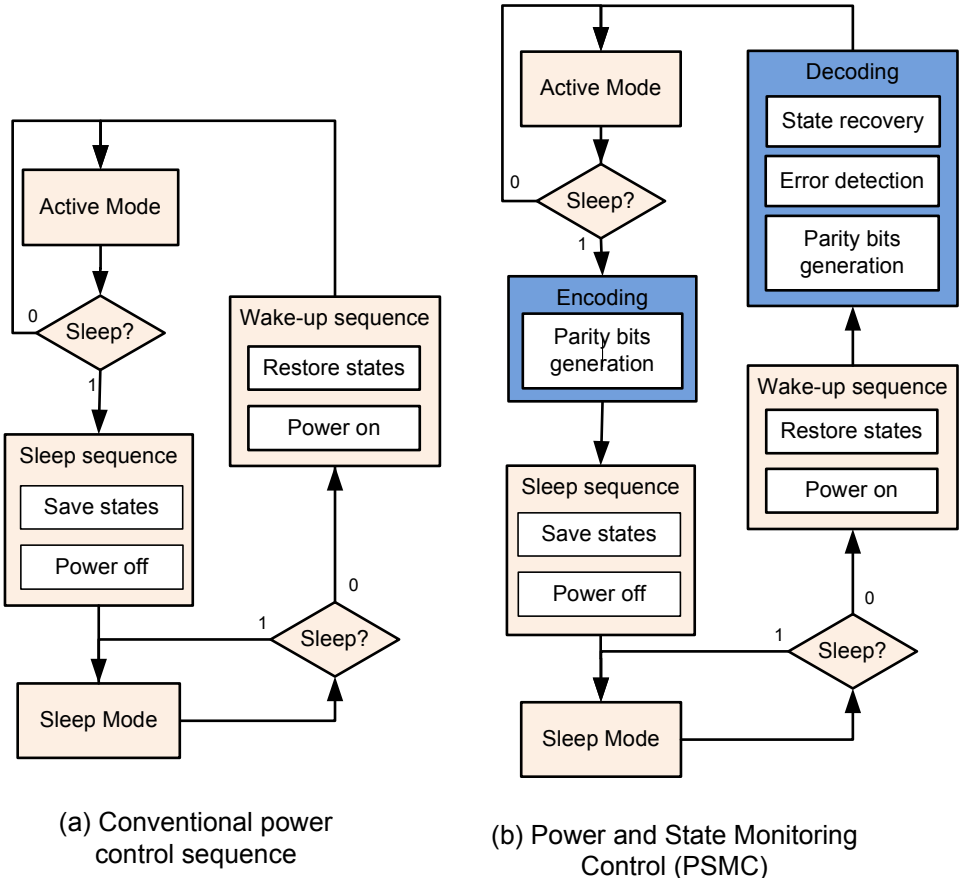


Figure 3.4: Control sequence (a) conventional (b) proposed Power and State Monitoring Controller (PSMC)

for drowsy logic the supply voltage is scaled down, then the circuit enters the sleep mode. When signal 'sleep' is '0' it starts the wake-up sequence, for power gating design it includes turning on the power transistors and restoring the circuit's states when the power supply become stable while for drowsy logic the power supply is restored and the circuit enters active mode. Figure 3.4 (b) shows the power and state monitoring control (PSMC) that is integrated into the conventional sleep control flow, where parity bits are generated and stored before the sleep sequence, and the circuit's states are checked against the stored parity bits after the wake-up sequence.

Figure 3.5 shows the timing diagram of the power and state monitoring controller, where 'sleep' is the external active high signal that triggers the sleep and wake-up sequence: when 'sleep' is set high, the controller isolates the outputs of the power-gated circuit; signal 'encode' is set high for a number of clock cycles depending on the length of the scan chains to enable the generation of parity bits; clock is stopped when encoding is finished; Signal 'retain' is set to low to transfer data from the flip-flop's slave latch to its retention latch and active low signal 'reset' is set to '0' to clear the content of the slave latch before powering off. The wake-up sequence is: when external signal 'sleep' is set low, the power controller first turns on the power then de-asserts 'reset'; retain is set

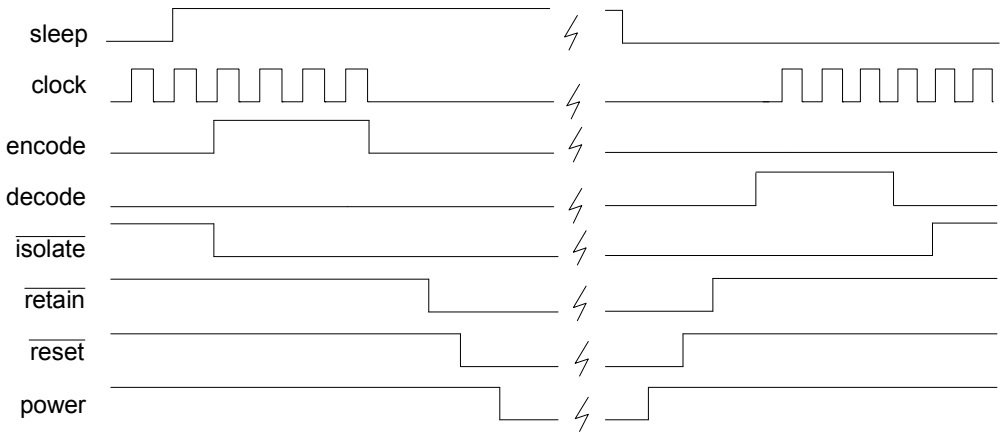


Figure 3.5: Power and State Monitoring Control timing diagram

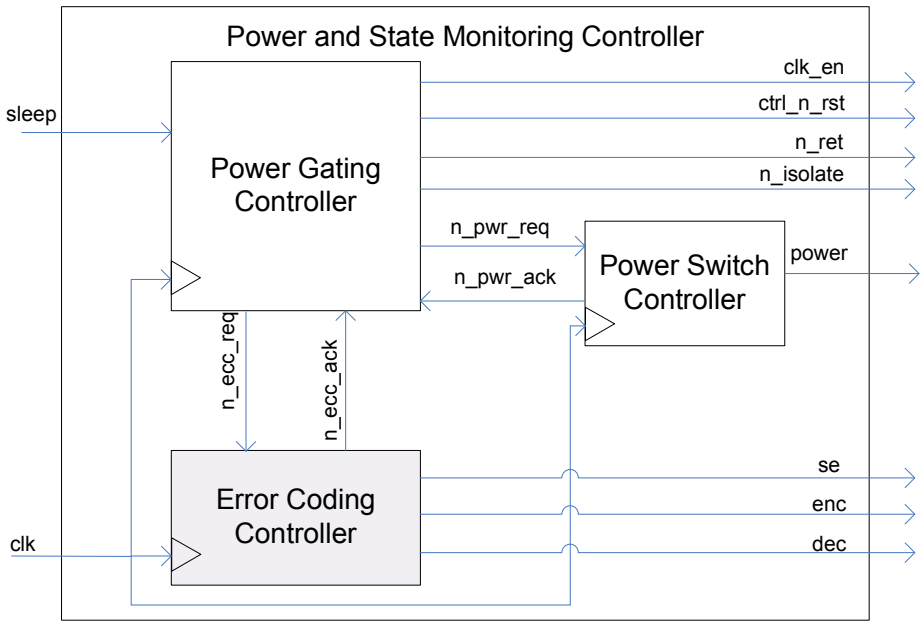


Figure 3.6: Power and State Monitoring Controller block diagram

high to transfer data back from the retention latch to the flip-flop's slave latch; the clock is enabled and the signal 'decode' is set high for number of clock cycles depending on the length of the scan chains to check the parity bits; at the end, output isolation is lifted and circuit returns to functional mode. The schematic of power and state monitoring controller is shown in Figure 3.6: the power-gating controller controls clock, reset, output isolation and saving/restoring circuit states; the power switch controller controls power switches; and the error coding controller is in charge of state monitoring and recovery. These controllers communicate using handshaking signals, so different coding schemes can be used without changing the power gating controller.

Figure 3.7 shows example implementations of state machine transition for (a) power gating control and (b) scan-based ECC control using Verilog. There are six main stages in power-gating control flow as shown in Figure 3.7.(a):

1. In active mode, the controller waits for sleep request. Right after sleep request is received, it then sends ECC generation request to ECC controller and waits for the acknowledge signal.
2. The standard sleep sequences before turning off power switches.
3. It sends power-down request and waits for the circuit to be powered off.
4. It powers up the circuit when receiving wake-up request.
5. The standard wake-up sequences after turning on power switches.
6. Error detecting and correcting using stored ECC code.

Only stage 1 and stage 6 are created for state monitoring and recovery; the rest of stages are identical to standard state retention power gating design.

3.1.3 Step 3: Synthesis Flow

Figure 3.8 shows the design synthesis flow for incorporating the proposed state monitoring and recovery method using a conventional design flow. It has three inputs, which includes the original design, the scan chain configuration file, and templates for SMRB (Figure 3.1) and PSMC (Figure 3.4). The synthesis flow consists of four main steps: it first inserts scan chains into the design through Design For Test (DFT) tool; it then instantiates the SMRB and PSMC through the template and the configuration file, and finally the design is synthesised using the RTL synthesis tool. Synopsys DFT Compiler and Design Compiler are used in this case study.

3.2 Case Study 1: FIFO

In this case study, the proposed method is used to improve the reliability of an FIFO circuit. The design is functionally validated using the Xilinx Virtex-5 FPGA on Xilinx ML505 evaluation board shown in Figure 3.9. Although there is no power switching and scan chains insertion in FPGAs, the reliable power control sequence (Figure 3.4) is emulated and the scan chains insertion is done in RTL level using a Perl script. To verify the proposed method a 32x32 bit FIFO circuit is created as a case study because it has a high density of flip-flops. 80 scan chains (selected for demonstration purpose) are created in the FIFO with 13 flip-flops in each scan chain. The state monitoring block (Figure 3.1) uses both Hamming code and CRC code; these are chosen because of their effectiveness in error detection and correction and simple encoding and decoding circuitry (small circuit area overhead). The testbench setup is shown in Figure 3.10. There are five components: FIFO_A consists of a FIFO module using proposed reliable power-gated

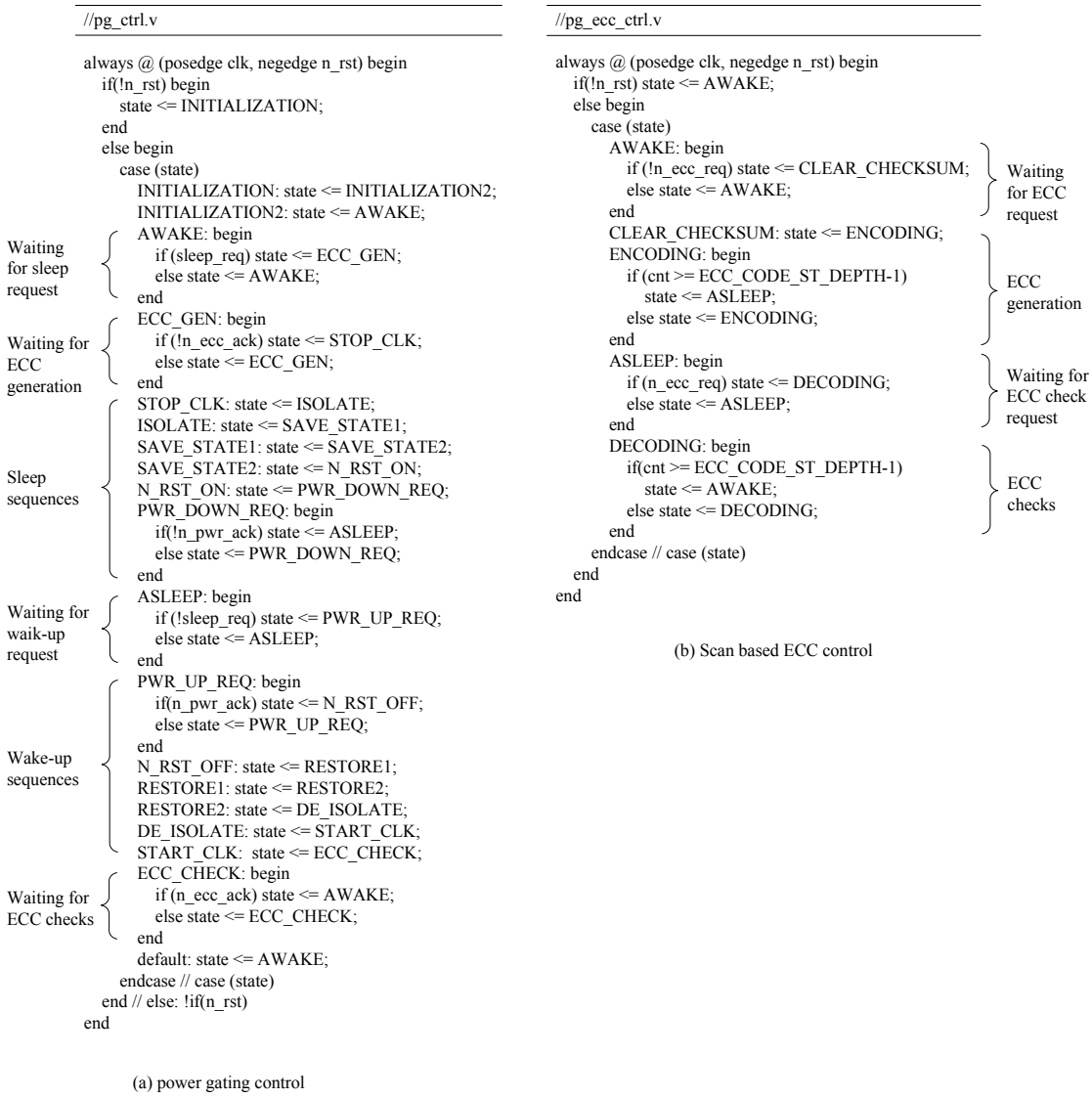


Figure 3.7: Example code snippet for (a) power gating control and (b) scan based ECC control

design and an error injection circuit; FIFO_B is the error-free reference FIFO module; Stimulus generates and writes random data to both FIFO_A and FIFO_B; Comparator reads the data from both FIFO_A and FIFO_B and compares them. Counters record each event when the errors are reported by FIFO_A and when the mismatches are reported by comparator. The test bench test sequence has five stages:

1. Reset both FIFO_A and FIFO_B to ensure they start in the same state.
2. Write to both FIFO_A and FIFO_B with the same random data.
3. Send sleep signal to FIFO_A and stop the clock to FIFO_B
4. Wait until FIFO_A is in sleep mode, then send wake-up signal to FIFO_A.
5. Read both FIFO_A and FIFO_B and compare the outputs.

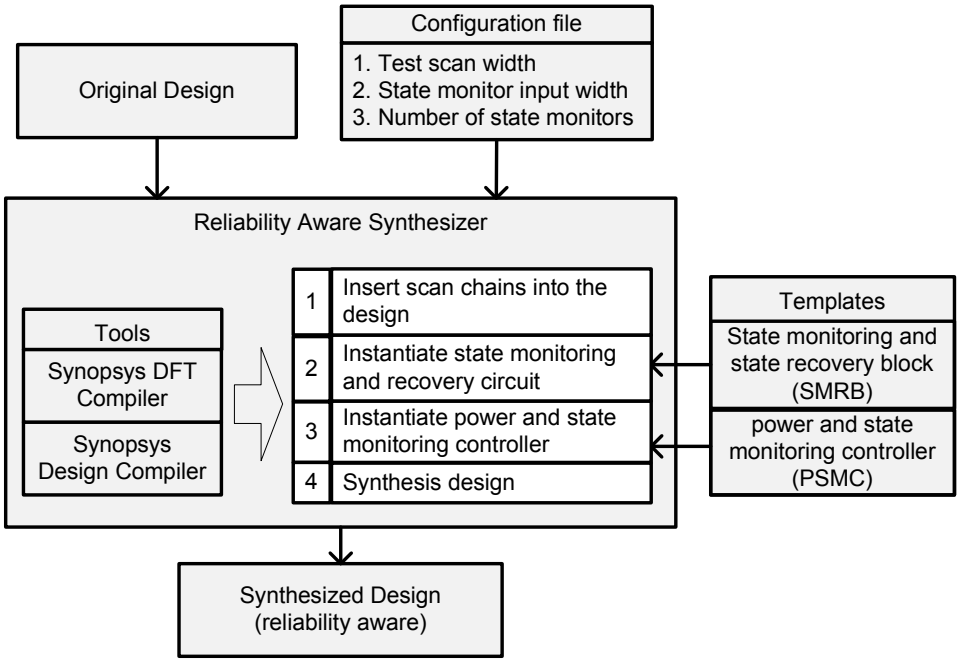


Figure 3.8: Proposed design flow for reliable state retention power gating design

Repeat the test sequence for a required number of times before sending the result to PC through RS-232 serial port.

3.2.1 Error Generation and Injection

Figure 3.11 shows a hardware error injection circuit used for injecting random errors. The error injection circuit consist of a column error injector and a row error injector, which target the error injection location. Each fault injection cycle consist of two stages: It first generates random errors by setting the column and row injector using linear feedback shift registers; then the injection circuit injects errors through scan chains by flipping the scan-out data and feed back into the scan-in ports. For example to inject a single error into the flip-flop in the 3rd row and 4th column shown in Figure 3.11, the row injector is set to '0010000' (top $\rightarrow$ down). Then set the column injector to '0001000' (left $\rightarrow$ right) and set the circuit in scan mode. The column fault injector shifts in the same direction (to the right) as the scan chains. When the column injector's output is '0' the fault injection is disabled by the 'AND' gates. After three clock cycles the column injector's output is shifted to '1', which enables the fault injection of the 4th column. Then the row injector will flip the 3rd bit of the column using 'XOR' gates. In the 4th clock cycle the error is latched into the circuit.

The error injection location can be described with the following equation:

$$Error\ location = \sum_{i=1}^{m} x_i \cdot \sum_{j=1}^{n} y_j \tag{3.1}$$

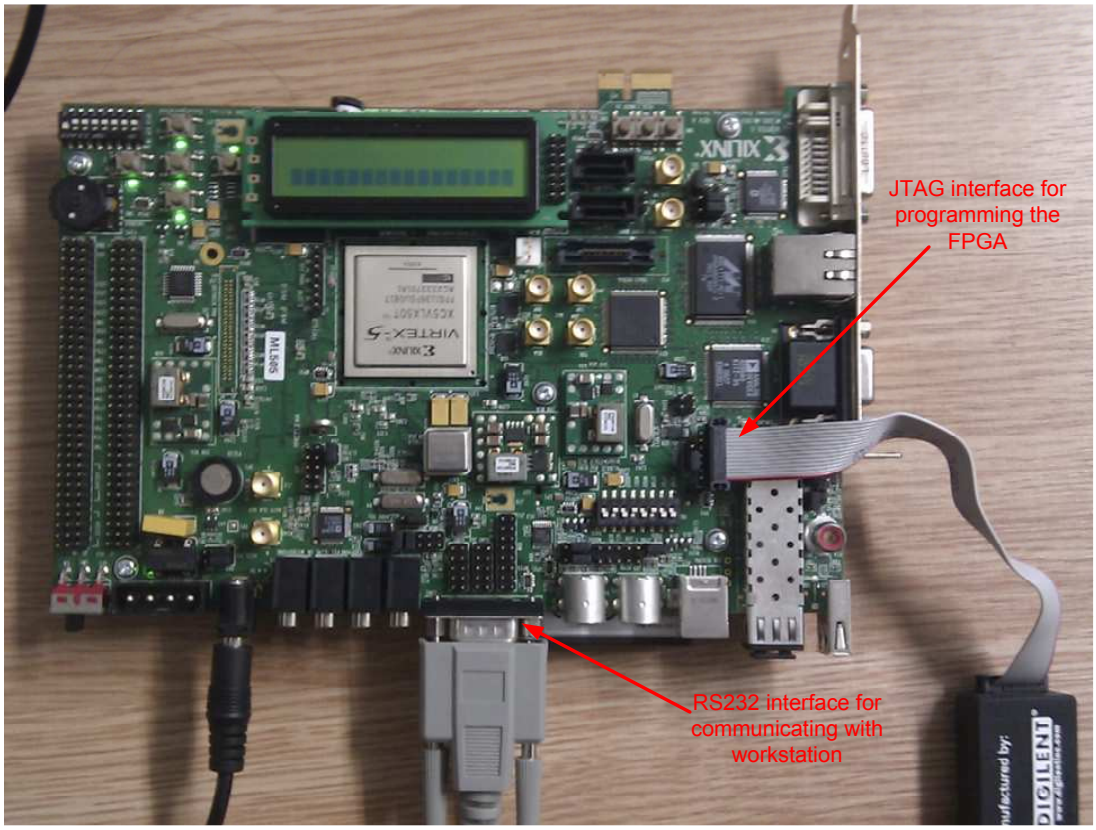


Figure 3.9: Xinlinx ML505 FPGA evaluation board for functional verification of the method

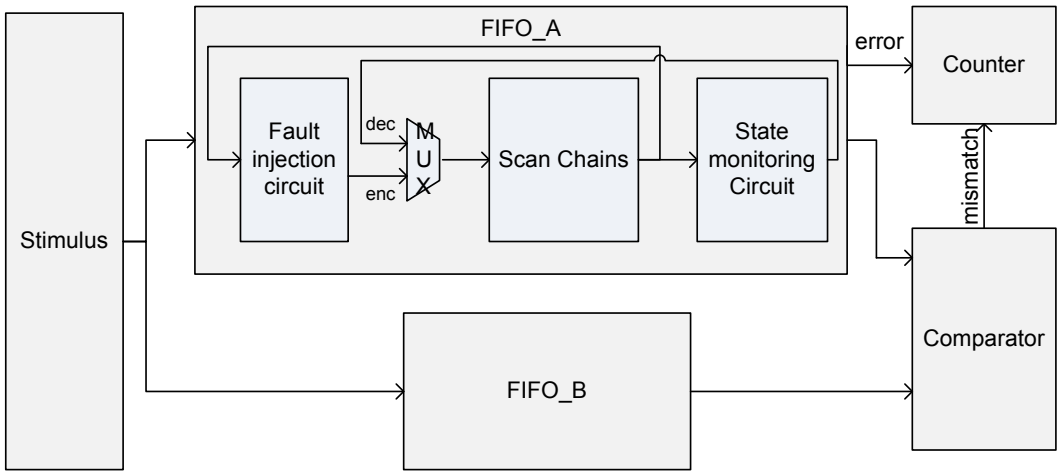


Figure 3.10: FPGA test bench for functional verification of the method

where x_i is the value of the bits in the column injector and m is the total number of bits in the column injector, y_i is the bits in the row injector and n is the total number of bits in the row injector. The error location is the $n \times m$ bit map produced by the bitwise multiplication of x_i and y_i, for each bit on the bit map 1 indicates an error and 0 indicates an error-free state.

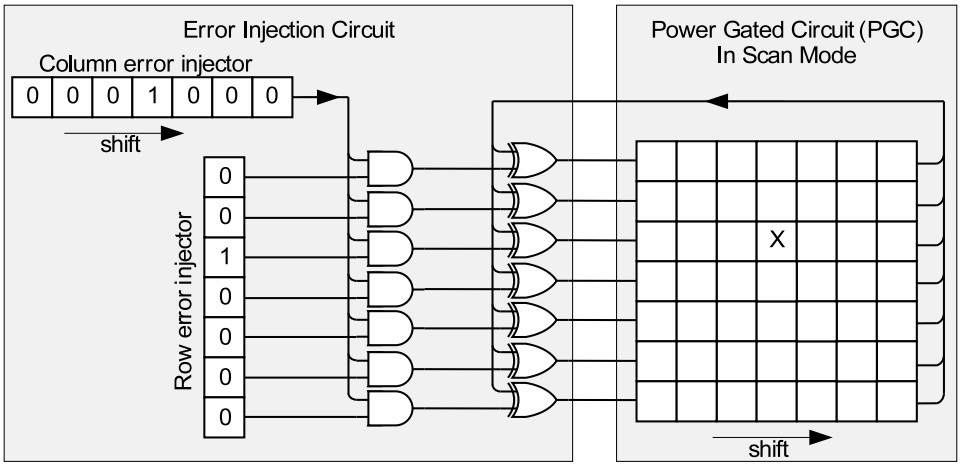


Figure 3.11: Error injection architecture.

Figure 3.12 shows four different error patterns that are generated for each test run of the experiment:

a) One error location is generated in the row(y) and column(x) injector; a single error is injected at location (x,y).

b) Set all column injectors to '1's and set the 1st bit of row injector to '1'. In each clock cycle shift the row injectors downwards, so that one error per scan slice is injected in each row.

c) The beginning and the end of error location are generated in both row(y_1 to y_2) and column(x_1 to x_2) injector, so that errors are injected in the area of (x_1,y_1 to x_2,y_2).

d) Random bit patterns are generated in each row(y_1,y_2,y_2...,y_n) and column(x_1,x_2,x_3...,x_m), so that errors are injected in the location ($\sum_{i=1}^{m} x_i \cdot \sum_{j=1}^{n} y_j$).

Figure. 3.13 shows Linear Feedback Shift Registers (LFSR) of length 'n' are used to generate random locations for the error injector. For the error pattern shown in Figure 3.12.(a) the single error location in both column and row injector is indicated by the value of the column and row LFSR respectively; if the length of column injector is 'm', then the length of column LFSR is $n = log_2(m)$. Similarly the length of row LFSR can be decided. LFSR is not needed for the error pattern shown in Figure 3.12.(b), because this pattern is not random. For the error pattern shown in Figure 3.12.(c) the beginning and the end of error location in column injector are indicated by the value of 2 LFSRs, same rule applies for the row injector. For error pattern shown in Figure 3.12.(d) the bit locations on both column and row injector are mapped from the column and row LFSR, and the length of LFSR is equal to the corresponding injector.

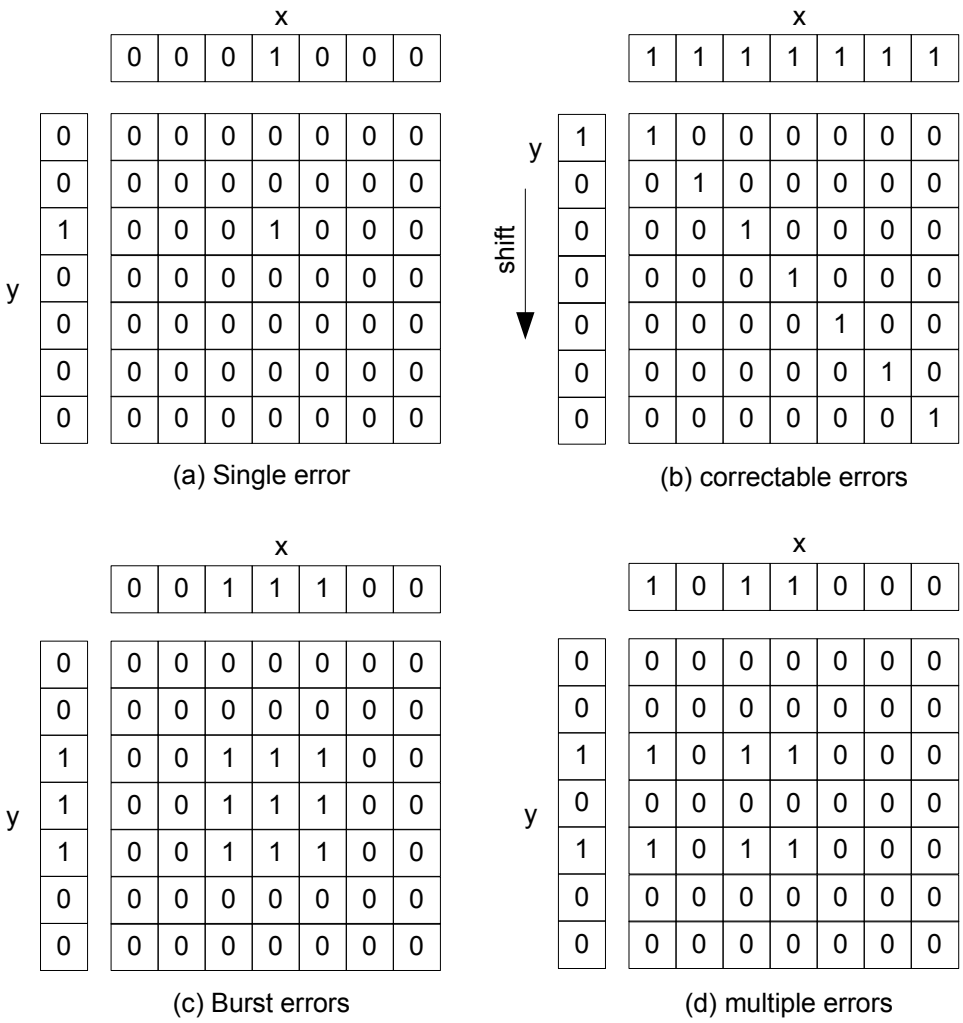


Figure 3.12: Error injection patterns.

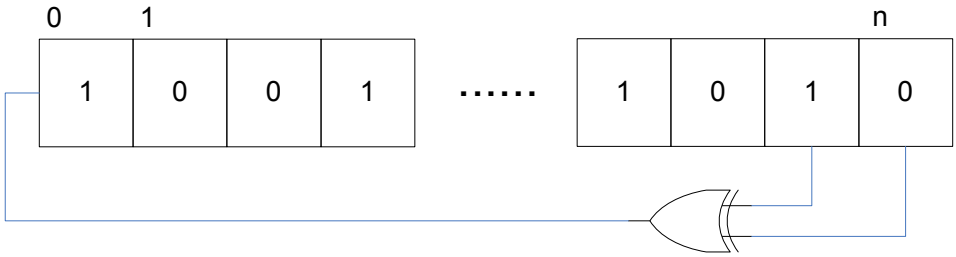


Figure 3.13: Linear Feedback Shift Registers

3.2.2 Experimental Results

To evaluate the effectiveness of the proposed method in improving the reliability of FIFO design, four experiments were performed using 1×10^6 test sequences, where each test sequence performs fault injection. In each experiment, one error pattern (from Figure 3.12) is used for fault injection. The result is shown in Table. 3.1, where the first column shows the number of injection runs for each experiment, the second column shows the number of test runs in which all injected errors are corrected using Hamming

code, the third column shows the number of test runs where not all errors are corrected by Hamming code but are detected using CRC and the fourth column shows the number of test runs where the output of FIFO_A is different from the output of reference design FIFO_B. There are four key observations:

1. The result of the first experiment (Figure 3.12.(a)) is shown in the first row of Table 3.1; the error correction circuitry detected and corrected all single errors per test sequence and therefore no error was reported by FIFO_A. This was further verified by comparing the outputs of FIFO_A and FIFO_B using a comparator.

2. The result of the second experiment (Figure 3.12.(b)) is shown in the second row of Table 3.1; the error correction circuitry detected and corrected all errors per test sequence. This is because there is only one error in each column, each column contains one or more code word which mean there is no more than one error in each codeword, and Hamming code is capable of correcting one error per code word. Therefore no error was reported by FIFO_A which was verified by comparing the outputs of FIFO_A and FIFO_B.

3. The result of the third experiment (Figure 3.12.(c)) is shown in the third row of Table 3.1; injected errors are corrected in some test runs where only one error per column is injected due to the upper bound and lower bound of error locations in the row injector are the same. The uncorrectable errors are all detected by the CRC error detection circuitry, which was verified by comparing the outputs of FIFO_A and FIFO_B using a comparator.

4. The result of the last experiment (Figure 3.12.(d)) is shown in the fourth row of Table 3.1; injected errors are corrected in some test runs where only one error per column is injected due to single error location in the row injector. The uncorrectable errors are all detected by the CRC error detection circuitry.

To summarise, errors can be corrected using Hamming code if there is only one error per column and all uncorrectable errors are accurately detected due to the fact that the high error coverage of CRC-16 code. Only certain error patterns can escape CRC-16 error detection and the fault injection method used in this experiment does not cover those pathological error patterns, so that CRC error correction can correctly detect all injected errors. More rigorous experimental coverage of error injection to cover the full state space is described in Section 3.3.2.

3.2.3 Trade-off Analysis

In this section the trade-offs are studied in terms of the area overhead of state monitoring circuitry, encoding/decoding time and power related to the implementation of two types

Table 3.1: FPGA fault injection, error detection and correction result

Injection pattern	Injection run	Corrected	Detected	Comparator output
single error	1000000	1000000	0	0
correctable error	1000000	1000000	0	0
burst error	1000000	55552	944448	944448
multiple error	1000000	3626	996374	996374

of coding (Hamming code and CRC code) with different scan chain configurations. In this section the terms 'latency' and 'encoding/decoding times' are used interchangeably. The design is synthesised using 120nm CMOS technology. The area overhead is generated from Synopsys Design Compiler. The gate level netlist of the power-gated design is simulated in the Cadence simulator, and the encoding/decoding power is calculated by Synopsys Prime Time PX. The circuit is clocked at 100MHz for demonstration purposes.

Table 3.2: Encoding and decoding circuit area overhead, power, latency and energy consumption for CRC-16 code with different scan chain configurations

32x32 FIFO, CRC-16 code, 120nm CMOS, clock=100MHz								
$Width$	$Length$	Area		Power(mW)		Time(ns)	Energy(nJ)	
		μm^2	%	encode	decode		encode	decode
4	260	73658	2.8	4.99	4.99	2600	12.97	12.97
8	130	73928	3.2	4.96	4.97	1300	6.45	6.46
16	65	74614	4.2	4.96	4.98	650	3.22	3.24
40	26	75762	5.8	5.13	5.17	260	1.33	1.34
80	13	78208	9.2	5.14	5.25	130	0.67	0.68

Table 3.2 shows the area, power, latency and energy when implementing the reliable power-gated FIFO using CRC-16 code. The first column shows the number of scan chains created, the second column shows the length of each scan chain, followed by the area of FIFO circuit, the 4th column shows the power consumption, the 5th column shows the timing performance, and the last column shows energy consumption. As the number of scan chains increases (from 4 to 80), the length of scan chains decreases from 260 to 13, and the encoding/decoding time decreases from 2600 ns to 130 ns. This is because encoding/decoding time is equal to the product of the length of scan chain and clock period, and longer scan chains lead to longer encoding/decoding time. This increase in the number of scan chains (from 4 to 80) results in area overhead (from 2.8% to 9.2%). This is because a higher number of scan chains requires additional state monitoring blocks (Figure 3.3 (a)) for encoding/decoding. Power consumption slightly increases (from 4.99 mW to 5.14 mW) with an increase in area; this is due to additional state monitoring blocks that consume extra power. The encoding/decoding energy decreases (from 12.97 nJ to 0.67 nJ) with the increase in the number of scan chains, because energy is the product of power and time; the number of scan chains increase the power by only 3% (relative difference from 4 to 80 scan chains) while latency

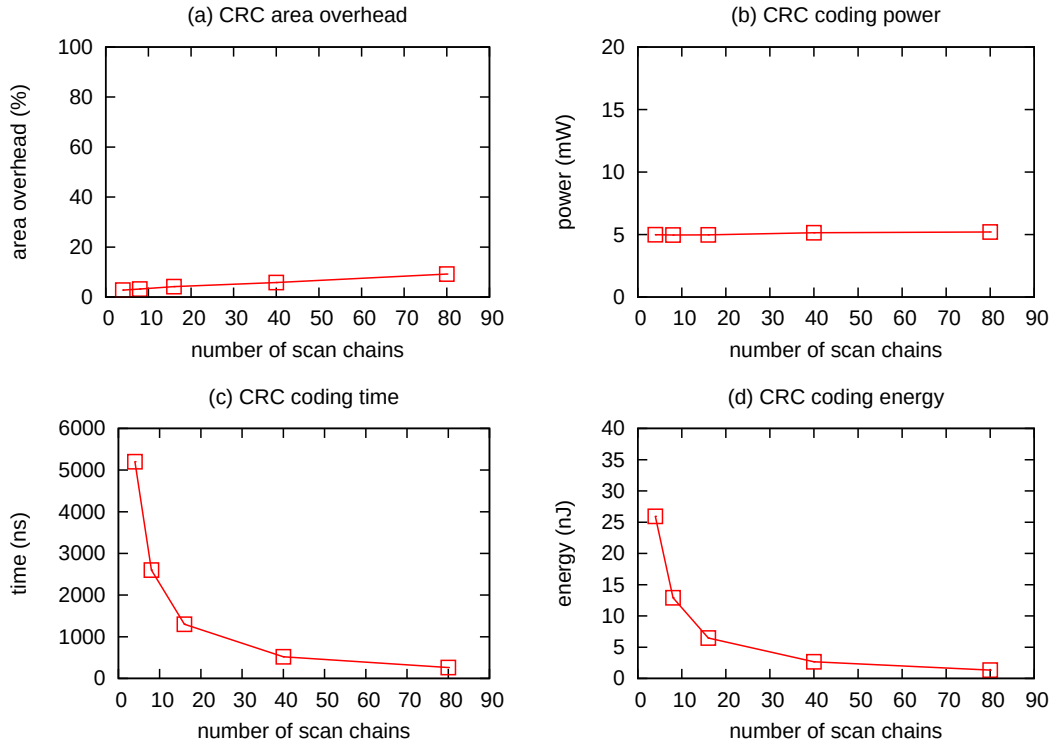


Figure 3.14: FIFO case study: CRC code implementation trades offs

decreases by 95%, which results in overall reduction in energy consumption. The CRC code implementation trade-offs are shown graphically in Figure 3.14.

Table 3.3: Encoding and decoding circuit area overhead, power, latency and energy consumption for Hamming (7,4) code with different scan chain configurations

32x32 FIFO, Hamming (7,4) code, 120nm CMOS, clock=100MHz								
Width	*Length*	Area		Power(mW)		Time(ns)	E(nJ)	
		μm^2	%	encode	decode		encode	decode
4	260	120594	68.4	6.76	6.72	2600	17.58	17.47
8	130	121552	69.7	6.91	6.86	1300	8.98	8.92
16	65	123303	72.1	7.11	7.00	650	4.62	4.55
40	26	126811	77.0	7.72	7.45	260	2.00	1.94
80	13	134141	87.3	8.43	8.05	130	1.08	1.05

Similarly Table.3.3 shows the area, power, latency and energy performance by using Hamming (7,4) code on reliable power gated FIFO. It shows a similar trend in terms of different scan chains configurations. When increasing the number of scan chains (from 4 to 80), the area overhead and encoding/decoding power increases (from 68% to 87% and from 6.76 mW to 8.43 mW respectively), but the encoding/decoding time and energy is reduced (from 2600 ns to 130 ns and from 17.58 nJ to 1.08 nJ respectively). The Hamming code implementation trade-offs are also shown in Figure 3.15.

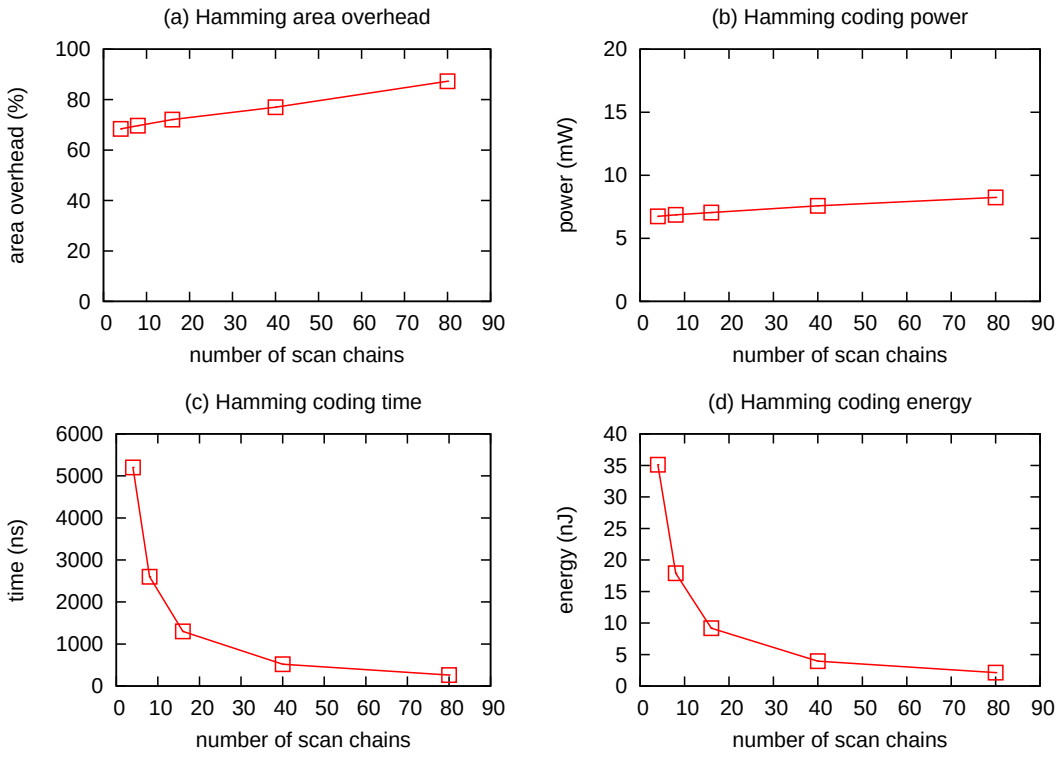


Figure 3.15: FIFO case study: Hamming code implementation trades offs

The area overhead of Hamming (7,4) code is significantly higher, at around 10 times more than the CRC-16 code (68% and 2.8% with 4 scan chains and 87% against 9.2% with 80 scan chains) as can be seen in Tables 3.2 and 3.3. This is because Hamming (7,4) code requires much larger parity bits storage than the CRC-16 code. The encoding/decoding power of Hamming (7,4) code is around 20% to 40% higher than CRC-16 code. The difference between power is not as significant as the area overhead because the majority of encoding/decoding power is due to the switching of scan chains, which is common in both implementations. The encoding and decoding time for both schemes are the same because latency is only affected by the scan chains length. The encoding and decoding of Hamming (7,4) code consumes around 20% to 40% more energy than the CRC-16 code due to Hamming (7,4) code's higher encoding/decoding power. For both the CRC-16 code and Hamming code, the encoding/decoding time and energy reduces significantly by increasing the number of scan chains at the cost of a relatively small increase in area and power.

Power-gated FIFOs using different Hamming codes are also implemented. Table 3.4 shows the area overhead and power consumption of different Hamming codes. The first column specifies implemented Hamming code, the 2nd column shows the number of scan chains inserted, the 3rd column shows the area overhead, the 4th column shows the power consumption, and the last column shows the maximum error correction ability of each implementation. As can be seen, the area overhead is minimum with Hamming (63,57) code but has least error correcting ability (1.59%). In general, area overhead can

Table 3.4: Encoding and decoding circuit area overhead, power, latency and correction capability of code to protect state for different Hamming codes

32x32 FIFO, Hamming code, 120nm CMOS, clock=100MHz							
code	$Width$	Area(μm^2)			Power(mW)		Correction
		FIFO	total	%	enc	dec	capability(%)
(7,4)	56	71628	132338	84.8%	8.21	7.84	14.3%
(15,11)	55	71628	101681	42.0%	6.52	6.34	6.67%
(31,26)	52	71628	88311	23.2%	5.89	5.82	3.23%
(63,57)	57	71628	82987	15.9%	5.64	5.62	1.59%

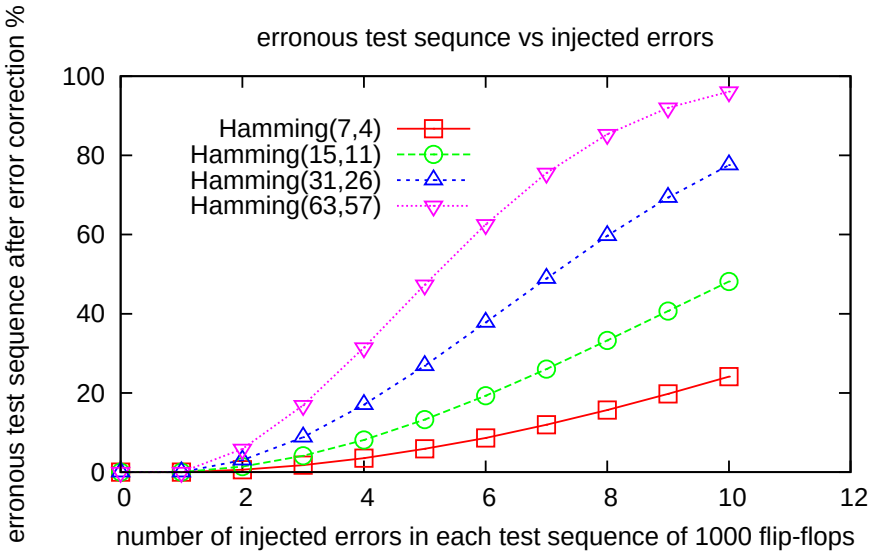


Figure 3.16: The probability of erroneous test sequence after error correction when multiple errors injected in each test sequence of 1000 flip-flops

be reduced (from 84.8% to 15.9%) using different Hamming codes at the cost of error correction ability that decreases from 14.3% to 1.59%.

The effectiveness of four types of Hamming codes used in this work were investigated. For this study, errors are randomly injected with even distribution in a test sequence of 1000-bits emulating 1000 flip-flops, up to 10 errors are injected in each test sequence and one million test sequences are simulated in total. The test sequence is then passed through four types of Hamming code implementation separately and the outcome is shown in Figure 3.16. As can be seen, Hamming (7,4) code has best error correction capability; it corrects all errors in 75.89% of test sequences when each test sequence contains 10 errors. Hamming (63,57) code has least error correction capability; it corrects all errors in only 3.9% of test sequences when each test sequence contains 10 errors.

3.3 Case Study 2: ARM Cortex-M0

For the second case study, an embedded microprocessor is chosen as a more complex test case, due to the need to support an additional software state recovery approach. ARM Cortex-M0 is an energy efficient and low cost 32-bit micro-controller [150], whose details are described in Appendix A. This section demonstrates how the proposed method can be incorporated in the embedded processor. A Cortex-M0 based processor platform is built and implemented in FGPA for this experiment, whose architecture is shown in Figure 3.17. Processor core Cortex-M0 is connected to AHB Lite bus as bus master, SRAM controller and ROM controller are connected to AHB Lite bus directly as bus slaves. GPIO (General Purpose Input and Output) and UART (Universal Asynchronous Receiver and Transmitter) are connected to APB bus which is connected to AHB bus as a bus slave. The UART is used for communication between the processor and the workstation. The programmer retrieves programs from the workstation through UART and writing them to the ROM. The interrupt controller senses the input buffer of UART and sends out interrupts to the processor core and notifies sleep controller when the input buffer is not empty. The sleep controller monitors the processor's operating mode (active or sleeping) using the "Sleeping" signal from the processor core and the "wake-up" signal from the interrupt controller. The processor core uses an Wait-For-interrupt (WFI) instruction as the software mechanism to request entry to sleep mode, while hardware interrupts wake up the processor.

Figure 3.18 shows the modified architecture for the ECC-protected Cortex-M0 based processor platform. Compared to the original system shown in Figure 3.17, three new components are created: State Monitoring and Recovery Block (SMRB), Power and State Monitoring Controller (PSMC), and error register. The State Monitoring and Recovery Block (SMRB) is connected to the scan chain of the processor core. The fault injection circuitry which is connected in the scan chains is used to validate the error detection and correction capability. The power controller is modified slightly to accommodate the state monitoring and recovery, referred to as Power and State Monitoring Controller (PSMC). The error register is inserted in the memory map to give the user program the access to the error information. Parity bits are generated by SMRB before the power down of the processor core and error detection and correction is undertaken after the power up of the processor core. The processor platform is implemented in several power domains to minimise the power consumption.

Figure 3.19 shows that the components of the embedded system are divided into three main power domains (PD). PD-1 is associated with functional blocks that are always-ON, which includes SRAM, error register, data bus, I/O peripherals, power and state monitoring controller (PSMC) and interrupt controller. PSMC and interrupt controller are required to be active during sleep mode to respond to external interrupt requests. In the case study SRAM, error register, data bus and I/O peripherals are not protected

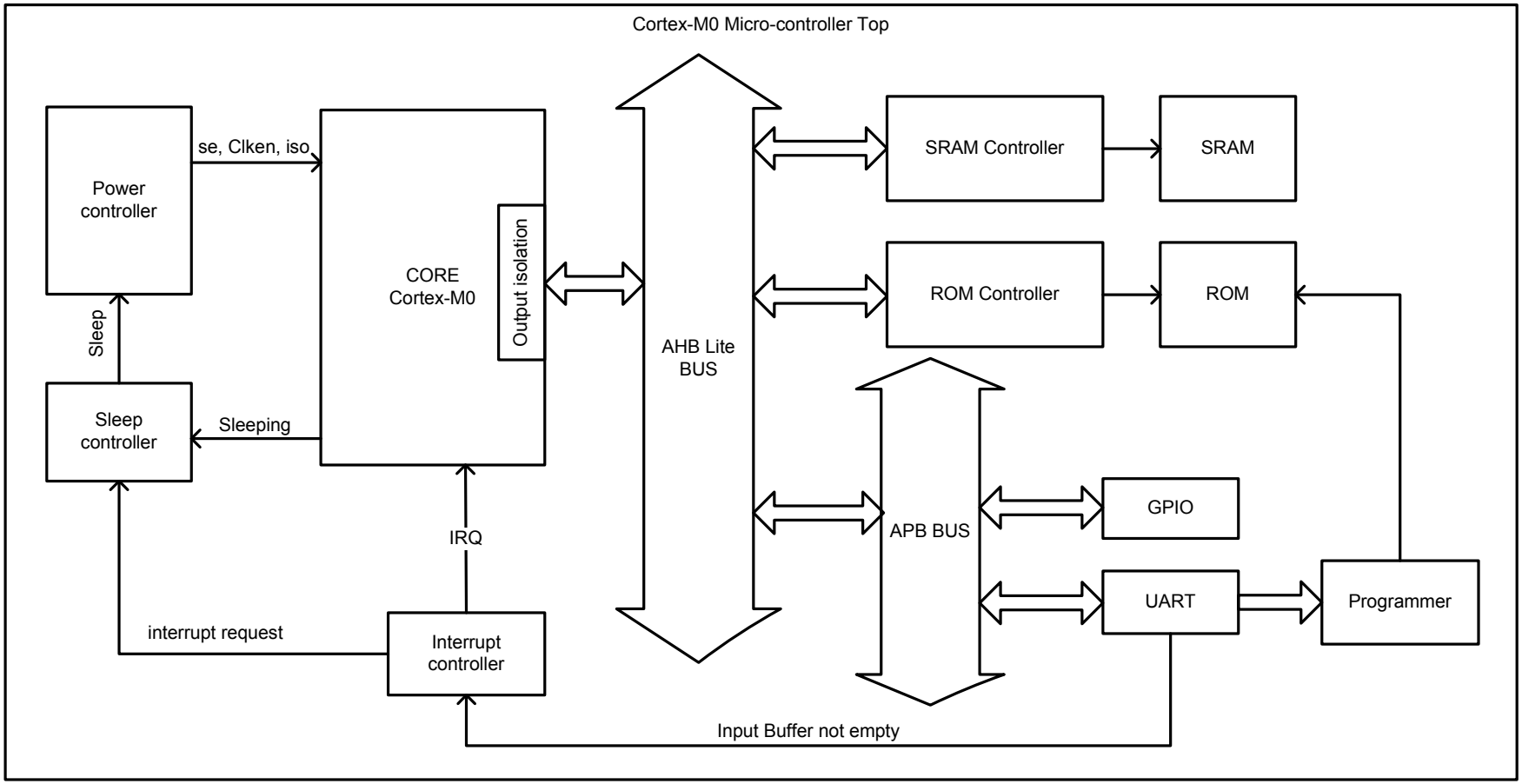


Figure 3.17: ARM Cortex-M0 processor platform block diagram

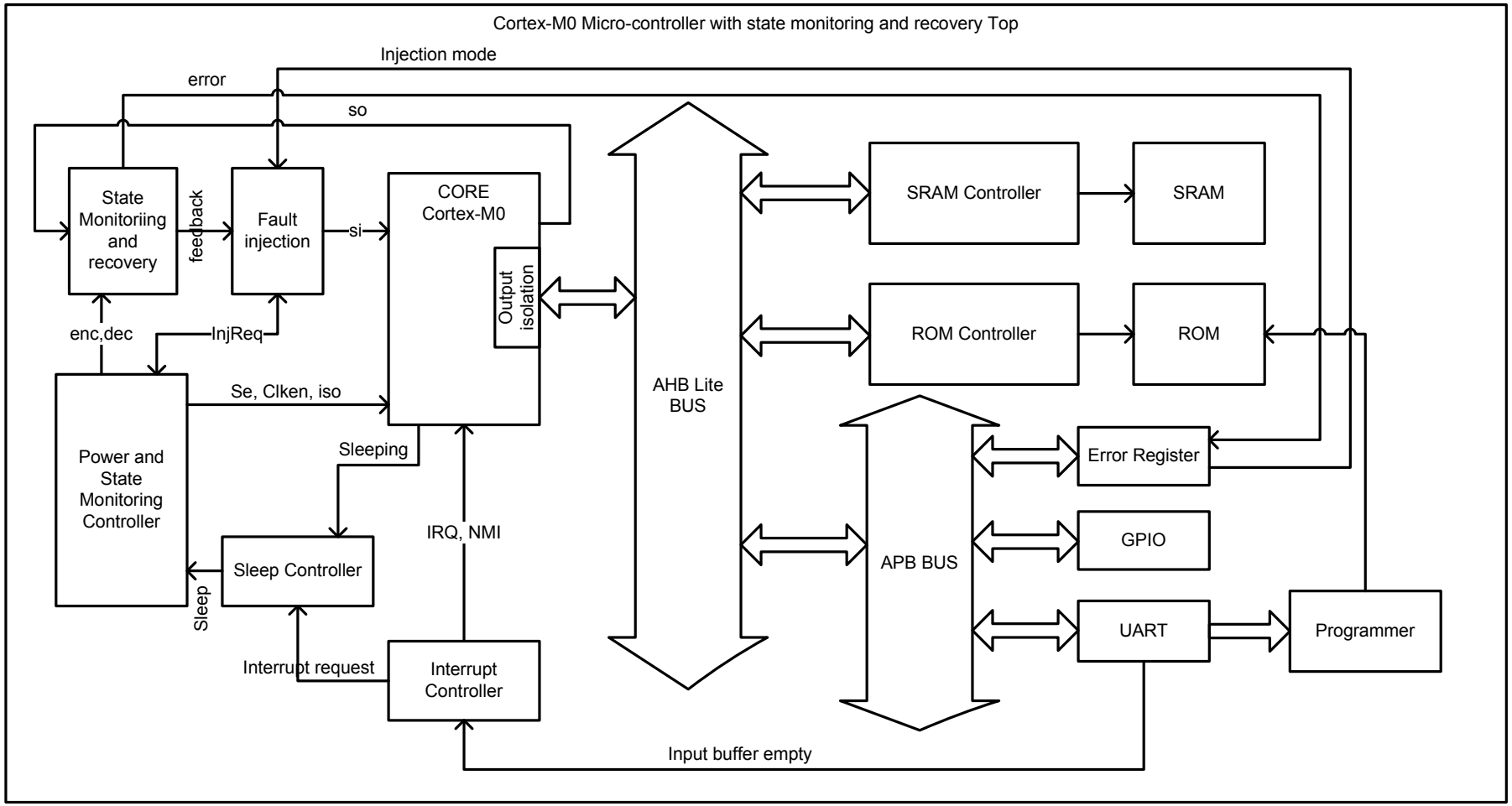


Figure 3.18: ARM Cortex-M0 processor platform with state monitoring and recovery block diagram

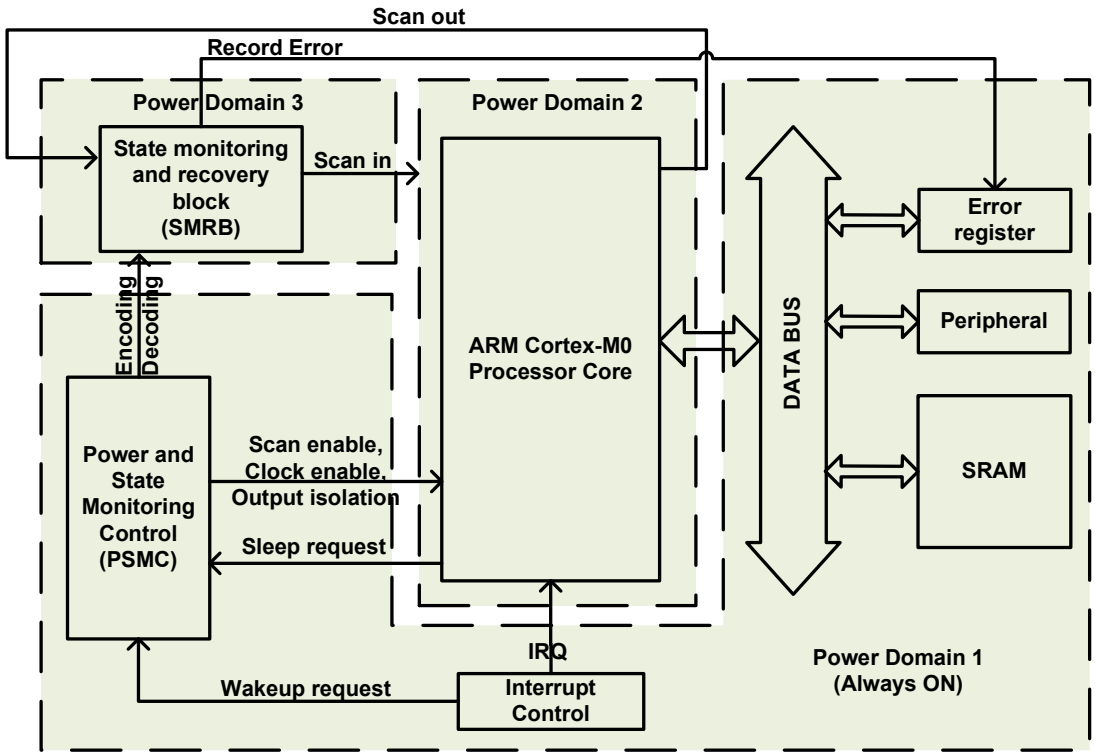


Figure 3.19: Block diagram of the proposed state monitoring and recovery technique using ARM Cortex-M0 as a case study.

by the SMRB. To guarantee their correct operation, they are placed in the always-on power domain (PD-1). PD-2 is associated with functional blocks that are powered-up only during active mode, which applies to the processor core. Finally, PD-3 is associated with the state monitoring and recovery block (SMRB), which is active only during encoding and decoding to ensure the processor's state integrity. PSMC controls the power up and power down sequence of the whole embedded system (Figure 3.19). It also controls the SMRB, which monitors the processor states during sleep mode and recovers the faulty states when an error is detected. All errors detected by the state monitoring block, but which are not correctable using hardware are recorded in the memory-mapped error register, so the processor has access to it through the data bus. The power domains are annotated onto the structrual RTL design and are described in detail using Unified Power Format (UPF) in Appendix. B.

In addition to the generic state monitoring and recovery control flow shown in Figure 3.4.(b), There are additional steps for incorporating software state recovery in the embedded processor, which are saving architectural states before encoding and restoring architectural states if hardware correction failed. Figure 3.20 shows the detailed control flow, the wake-up sequence is initiated by the "Peripherals" which sends an interrupt request through the "Interrupt Controller" to the PSMC. The PSMC first turns on the power supply of PD-2 and PD-3, and then initiates the state "Decoding" through the SMRB. SMRB re-generates the parity of the processor states and compares it with the

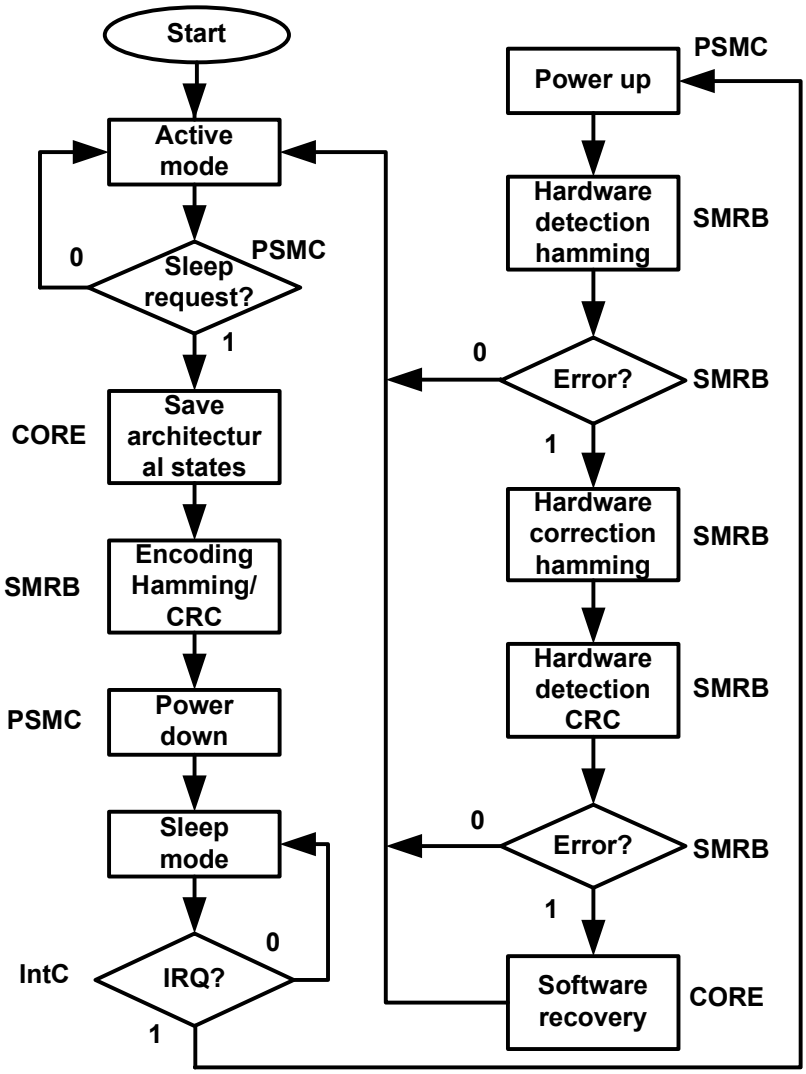


Figure 3.20: Control flow of error detection and state recovery mechanism.

stored parity bits (generated before going to sleep mode). In the case of a mismatch (error detection), the following sequence of events occur: Firstly, an attempt is made to recover the states through Hamming code, and if successful, the corrected states are fed back through the processor scan-in port. Secondly, the state integrity is checked by comparing the parity generated by the CRC code of the corrected states (through Hamming) with the one generated before going to sleep mode. In the case of a mismatch, which indicates that the hardware error correction is not successful, this event is registered in an error register. Now the processor core is in an unreliable state so it must be reset; the reset handler will check the reset entry reason by examining the error register, and if it is due to state corruption it will branch to the software recovery routine before servicing the wake-up event. The detail descriptions of software recovery firmware is presented in Appendix C. After the processor's architectural states are recovered, the processor is redirected to the appropriate instruction for the active mode of operation.

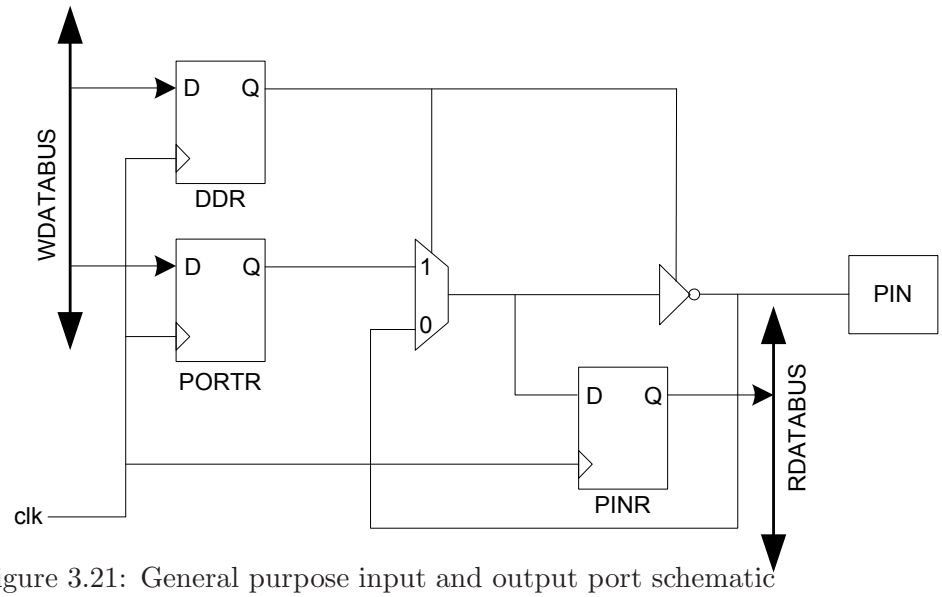


Figure 3.21: General purpose input and output port schematic

3.3.1 Processor System Components

This section shows the detailed implementation of processor system components. General purpose input and output (GPIO) is used in most processor platforms to provide digital control signals or software reconfigurable interface. The schematic of the 1-bit GPIO unit is shown in Figure 3.21. The port register (PORTR) takes the data from the data bus and make it available to the output pin. The pin register (PINR) samples the data from the input pin which then can be read by the data bus. The data direction register (DDR) controls the function of a particular pin, which can be set by the data bus. When DDR is set to '0', the tristate buffer disables the output and the multiplexer directs the input data to the pin registers; the pin becomes an input pin. When DDR is set to '1', the multiplexer selects the data from port register and the tristate buffer is enabled; the pin becomes an output pin.

The universal asynchronous receiver and transmitter (UART) is implemented to provide simple communication between the workstation and the processor using the RS-232 standard. There are two major functions; first it helps the programmer to download binary program code to the system ROM, and secondly it provides the remote access console interface. Figure 3.22 shows the schematic of the UART unit. It has a receiver which covert bit-stream from the RS-232 interface to bytes and stored in receiving buffer which can then be read by the data bus. When new data are read, the receiver sends an interrupt signal to the interrupt controller that notifies the processor core. The transmitter takes the data from the transmitting buffer and converts it into bit-stream which can then be sent to the workstation through the RS-232 interface.

The ARM Cortex-M0 has built-in support for power gating. It has two operational mode active mode and sleep mode; in sleep mode, the processor core can be safely powered off. A sleep controller is implemented to control the sleep and wake-up sequence. The

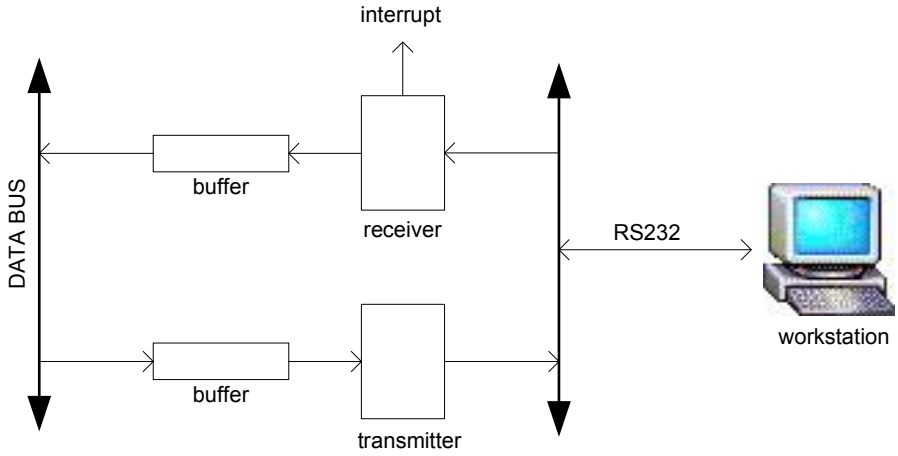


Figure 3.22: Universal asynchronous receiver and transmitter schematic

sleep controller is initialised to the "ACTIVE" state from reset where the processor core is running a normal program. The WFI (wait for interrupt) instruction signals the processor to stop program execution and enter sleep mode where it is safe to stop the clock, save the states and turn off the power. Figure 3.23 shows the state machine of the sleep controller. The processor core sets The "Sleeping" signal of the processor core is set high to indicate that the processor core is ready to be powered down. The sleep controller enters the "TRANSITION TO SLEEP" state and the sleep controller signals the power controller to turn off the power. After the power down sequence is completed, the power controller sends an acknowledge signal to the sleep controller by setting "pwr_down_ack" high; the sleep controller enter the "SLEEPING" state. When the sleep controller senses the interrupt from the interrupt controller in "SLEEPING" state, it holds the interrupt signal and sends power-up request to the power controller to initiate the wake-up sequence. Then the power controller powers up the processor core, enables the clock and sends an acknowledge signal to the sleep controller by setting "pwr_up_ack" high. The sleep controller goes back to the "ACTIVE" state and passes the interrupt signal to processor to process the pending interrupt.

The interrupt controller is used to coordinate the interrupt events from the devices which require the attention of the processor core. Figure. 3.24 shows the schematic of interrupt controller. In this system there are two devices that require interrupt services: error register and UART; these are shown on the right hand side of Figure. 3.24. When a new byte of data is received by the UART, it sends an interrupt requesting processor core to read the newly arrived data. When the error register records the uncorrectable error event, it sends an interrupt and asks for immediate attention to cope with the error (roll back to the clean state through software checkpoint), and the interrupt controller gives the error register a higher priority over UART. When an interrupt is received, the interrupt controller first checks the operation mode of the processor core by checking the state stored in the sleep controller. If the processor core is in active mode, the interrupt controller sends the interrupt to the processor core; if the processor core is

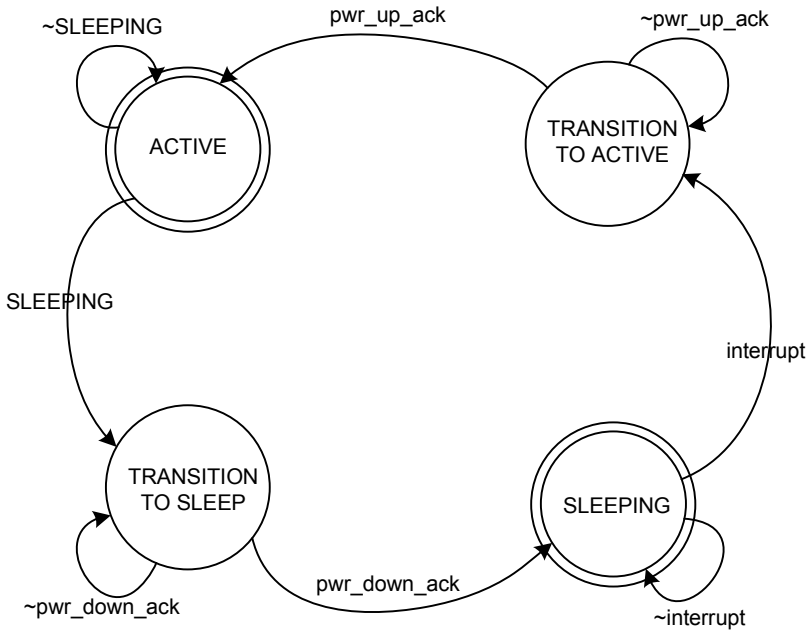


Figure 3.23: Sleep controller state machine

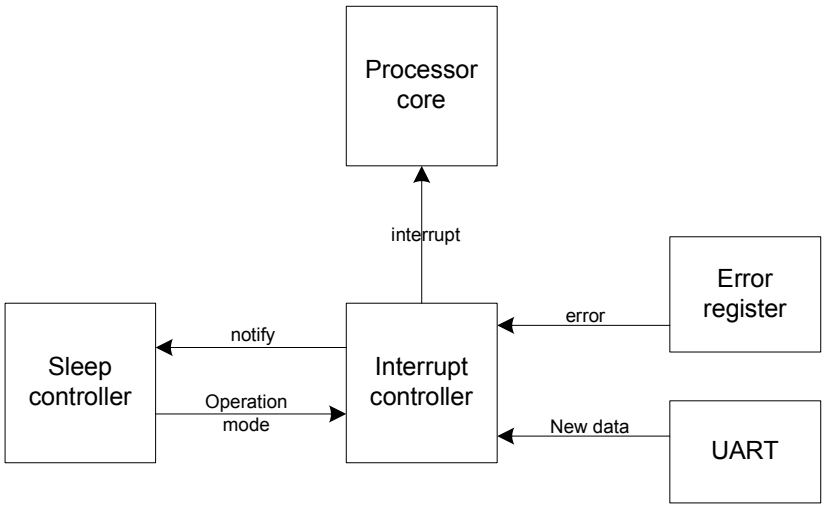


Figure 3.24: Interrupt controller

in sleep mode, the interrupt controller holds the interrupt event and notifies the sleep controller to wake up the processor core. As soon as the processor core is back to active mode, the interrupt controller sends the interrupt to the processor core.

Figure 3.25 shows the state machine of the interrupt controller. It starts in state 'S0' waiting for the interrupt event from peripheral devices. When interrupt is detected, it goes to state 'S1' and checks the processor states. If the processor is not in sleep mode, it goes to state 'S3' and sends the interrupt to processor. If the processor is in sleep mode, it goes to state 'S2' and notifies the sleep controller to wake up the processor, and waits for the processor to be powered up. When the processor is powered up, it goes to state 'S3' and sends the interrupt to the processor.

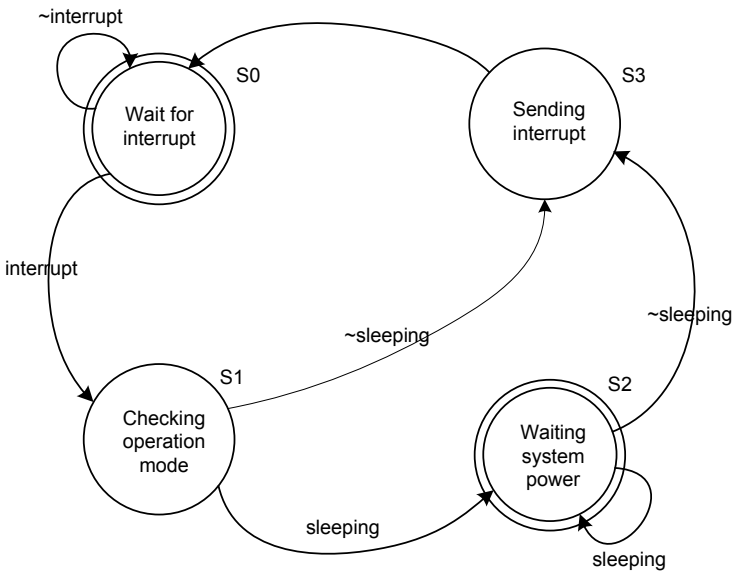


Figure 3.25: Interrupt controller state machine

Figure 3.26 shows an example of the text file containing a program that is transmitted to the system through UART. When the processor system is in programming mode, the programmer downloads the program through UART, converts it to hex numbers and writes it to the program memory "ROM". For example, the programmer reads data from the UART receiving buffer. When the character 'p' is detected, the programmer converts the following 8 characters from 8-bits ASCII code to 4-bits binary numbers, then it combines 8 sets of 4-bit binary numbers into 32-bits word to be written into the "ROM".

Figure 3.27 shows the memory map for the processor platform. The private peripherals are the control registers of the processor core, such as system timer register, interrupt control register and system control register [150]. The error register holds the error flag and controls the error injections. The UART_TXD and UART_RXD are UART transmitting and receiving buffer entry. The GPIO_DDR and GIPO_DATA are GPIO data direction registers and data registers. SRAM is used for storing intermediate data and program stack. ROM holds the program instructions.

3.3.2 Error Generation and Injection

Error rate is application, environment and implementation dependent [151]; this is why a wide range of bit error rate is considered in this work, which is from 10^{-12} to 10^{-1} errors per bit-hour. When considering soft errors, a study has reported that the error rate is in the range of 10^{-12} to 10^{-7} errors per bit-hour [152]. It is observed in a recent publication [153] that soft error rate (SER) is exponentially increasing mainly due to reduction in critical charge and supply voltage. In addition, power supply fluctuations and rush currents may also induce clustered errors, which may lead to higher error rates.

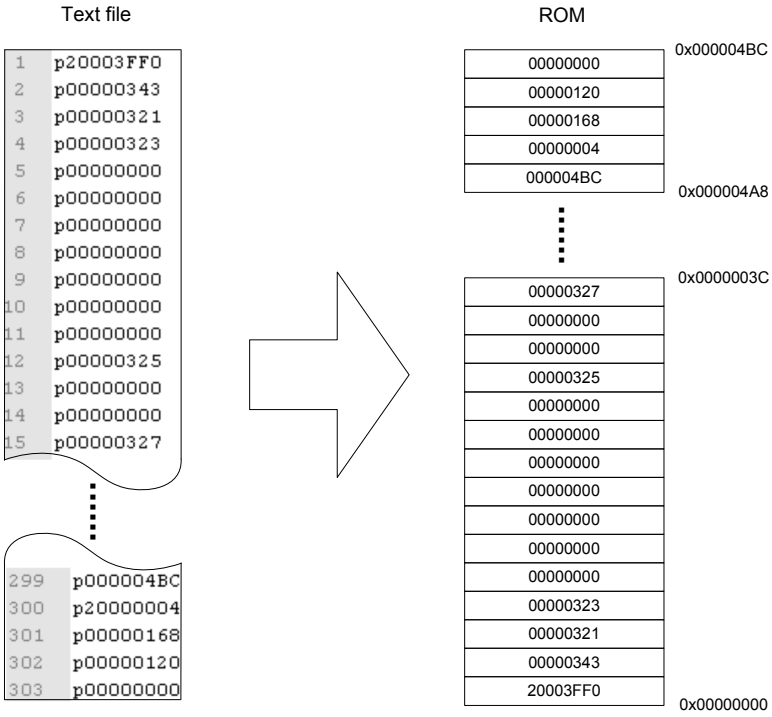


Figure 3.26: Programmer converts text file to data in ROM

This is why error rates up to 10^{-1} per bit-hour and clustered errors are considered in this work, represented by bit error rate (BER). This is used to develop a robust system for state integrity, which is applicable for future technology nodes.

In Section 3.2.1, a simple LFSR was used to generate random error locations for fault injection, which provides limited sets of error patterns. In this work, the Mersenne Twister algorithm [154] is used to provide higher randomness (longer repeating period) for error injection. Radiation-induced soft errors are evenly distributed and the available soft error injection model uses uniform distribution [155]. The proposed method also protects the embedded processor from clustered errors caused by power supply fluctuations and rush currents. Clustered errors are generated using a model presented in [156]:

$$P_i = \gamma \cdot \sum_{j=1}^{N_{ini}} \left(\frac{1}{d_{ij}} \right)^{\alpha} \tag{3.2}$$

where P_i is the error probability of each node, d_{ij} is the distance between two flip-flops i and j, α is the clustering factor, and higher α implies higher clustering. α was varied from 0 to 2 in these experiments, where 0 represents uniform distribution of errors (Figure 3.29a) and $\alpha > 0$ represents clustered errors (Figure 3.29b), N_{ini} is the number of initial errors and γ is the normalisation factor, which is calculated using the following equation:

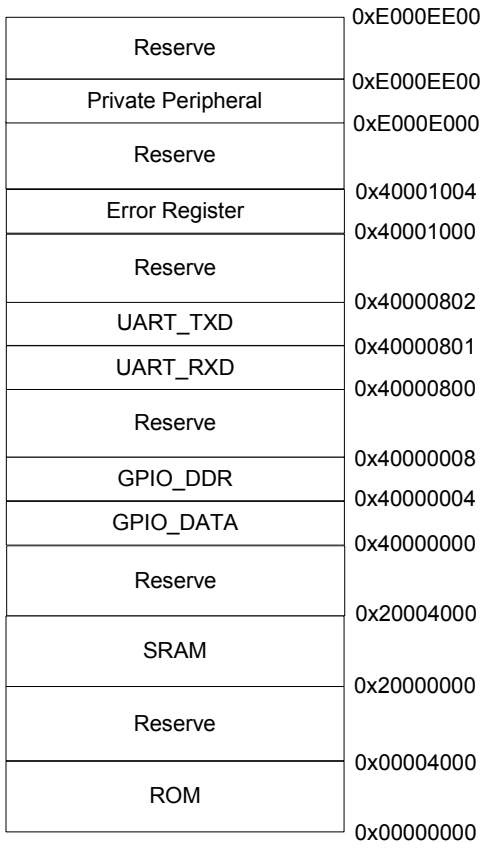


Figure 3.27: Processor Memory Map

$$\gamma = \frac{\epsilon_{tg}}{\frac{1}{N_{rst}} \cdot \sum_{i=1}^{N_{rst}} \sum_{j=1}^{N_{ini}} \left(\frac{1}{d_{ij}}\right)^{\alpha}} \tag{3.3}$$

where ϵ_{tg} is the target error rate, and N_{rst} is the number of uncorrupted bits after initial error injection. The number of clusters is decided by the initial injected errors N_{ini}, given a specific error rate for a design, N_{ini} is inversely proportional to the number of errors per cluster. In the case-study, N_{ini} is chosen to be 10% of total errors.

The error injection method is shown in Algorithm 1 and the example implementation is shown in Figure 3.28, which consists of two stages. First initial errors N_{ini} are evenly injected (step-1 to step-7), which is kept at 10% of target error rate ϵ_{tg}. Next, in the second phase, the error probability P_i of all the flip-flops in the system is recalculated, and clustered errors are injected (step-8 to step-14).

The C code implementation for generating cluster error distribution is shown in Figure 3.28. The injection stage is straightforward given the bit error rate of each node, only the stages that calculate the error probability are shown here. The first stage calculates the error-occurring probability for each bit in relation to the cluster centres (initial injection). The second stage scales the probability by the normalisation factor γ.

Algorithm 1 Method for Error Injection

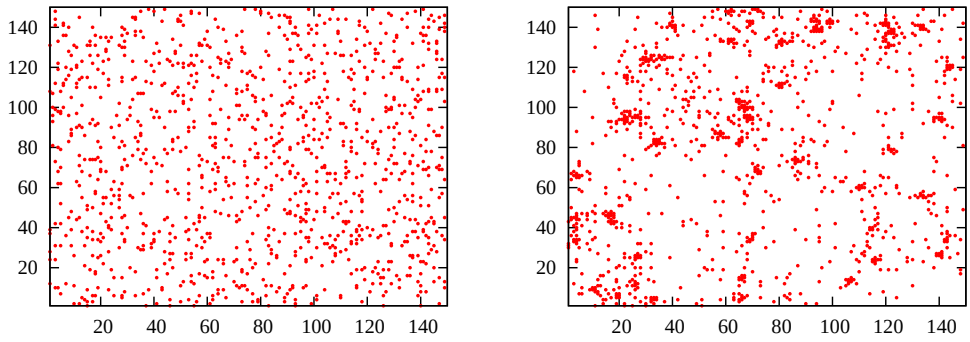
for $i = 1$ to N **do**
  $P_{init} = \epsilon_{tg} \cdot \beta$
  // P_{init} *is the initial error probability*
  // ϵ_{tg} *is target error rate,* β *is the initial injection ratio*
  // N *represents the total number of flip-flops*
  $R_i \leftarrow$ Generate a random number
  if $R_i < P_{init}$ **then**
    Inject error at node i
  end if
end for
for $i = 1$ to N_{rst} **do**
  Calculate P_i
  // P_i *is calculated using Equation* (3.2) *and Equation* (3.3)
  $R_i \leftarrow$ Generate a random number
  if $R_i < P_i$ **then**
    Inject error at node i
  end if
end for

```
while (plist){
    for(i=0; i<sizex; i++){
        for(j=0; j<sizey; j++){
            if(!bit[i][j].err){
                x0=plist->pbit->cordx;
                y0=plist->pbit->cordy;
                dist = sqrt(((i-x0)*(i-x0)+(j-y0)*(j-y0)));
                errProb = gsl_sf_pow_int(1/dist,alpha);
                bit[i][j].errProb += errProb;
            }
        }
    }
    plist = plist->pnext;
}
sumErrProb=0;
for(i=0; i<sizex; i++){
    for(j=0; j<sizey; j++){
        sumErrProb += bit[i][j].errProb;
    }
}
avgErrProb=sumErrProb/(sizex*sizey);
ratioErrProb=avgErrProb/(ber*(1-INITIAL_ERRORS));
for(i=0; i<sizex; i++){
    for(j=0; j<sizey; j++){
        bit[i][j].errProb = bit[i][j].errProb / ratioErrProb;
    }
}
```

Calculates the error probability based on the distance from the cluster center

Normalize the error probability so it does not exceed the target error rate

Figure 3.28: Cluster error injection implementation



(a) Unclustered Errors $\alpha = 0$ (b) Clustered Errors $\alpha = 2$

Figure 3.29: Error distribution using error injection method shown in Algorithm 1

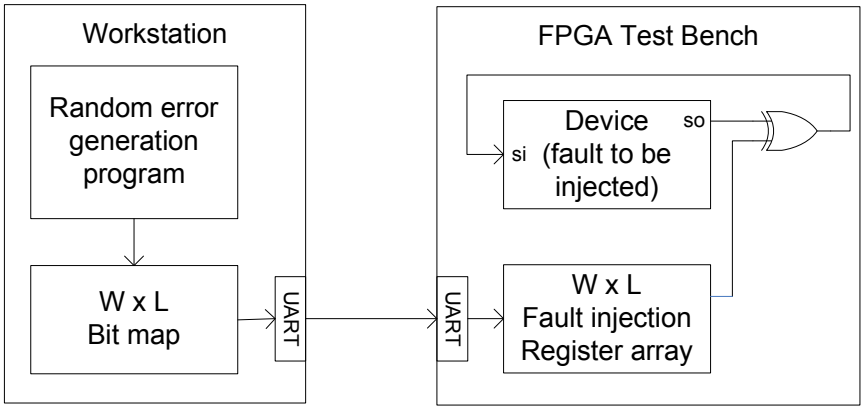


Figure 3.30: Hardware error injection architecture

These error distribution patterns are generated on a workstation in the form of memory maps indicating error locations. To inject errors at specific locations in the design, the memory map is transferred into the FPGA test bench through UART as shown in Figure 3.30, where fault injection registers hold the error location map, which is XORed with the states of the protected circuit through scan chains.

3.3.3 Experimental Results

Three experiments are performed to evaluate the effectiveness of the proposed method in improving the reliability of embedded processor implementations, in terms of hardware error detection, hardware error correction and software state recovery. CRC is well-known for its high detection capability, low area overhead and simple design [157]. An experiment is conducted to compare the detection capability of CRC-8 and CRC-16 under different bit error rates ranging from 10^{-1} to 10^{-6} errors/bit. Using the hardware setup shown in Figure 3.19, errors were injected evenly across all 768 flip-flops in the processor core for 10 thousand test runs at each bit error rate. The results are shown

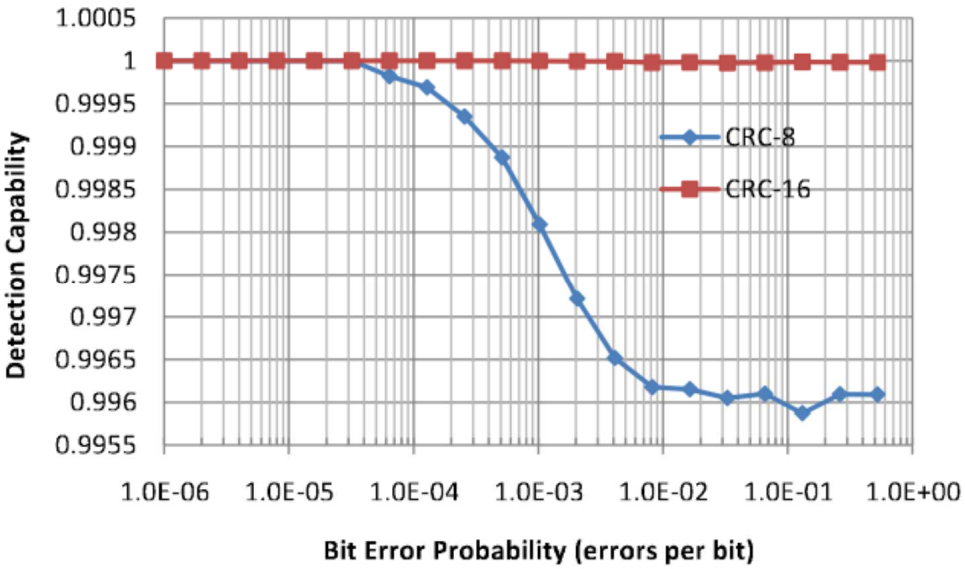


Figure 3.31: Detection Capability of CRC-8 and CRC-16 on Cortex-M0 as a case study

in Figure 3.31. As can be seen, both codes exhibit nearly 100% error detection capability when the bit error rate is less than or equal to 10^{-4} errors/bit. However, the detection rate of CRC-8 dropped down to 99.6% under 10^{-2} or higher bit error rates, while CRC-16 continued to detect nearly 100% of all injected errors. For this reason, CRC-16 is used in this work for hardware error detection. The polynomial used in this work is the same as the one used to encode data in USB 2.0 [158]; that is $(x^{16} + x^{15} + x^2 + 1)$. To enable faster execution a parallel-CRC implementation [159] is used.

The effect of radiation on 65-nm memory has been studied [160], where 44 upset events and 70 bit flips per Mbits are observed in 2.5×10^7 hours, 25% of which are multiple-bit upsets and always affect the adjacent bits. This implies that these multiple-bits upsets are caused by the same partical strike. The flip-flops in a circuit are not clustered as memory cells and therefore the chance of radiation event corrupting multiple flip-flops is much lower than corrupting multiple memory cells. If the bit error rate is λ and there are R flip-flops in the system, the unprotected system failure rate is:

$$System\ Failure\ Rate = \lambda \cdot R \tag{3.4}$$

Hamming code can achieve single error tolerance in each codeword and it can correct multiple errors in a circuit if these errors are not in the same codeword. It has small encoder and decoder area overheads and fast encoding and decoding. The number of times a state is checked affects state integrity of a storage element; state integrity of a flip-flop that is checked once a day is lower than the one checked every minute. This concept is called ECC scrubbing [161] and the checking period is termed as scrubbing interval. In this work scrubbing interval is equal to sleep duration. Assuming every

particle strike only causes a single bit upset and at sleep frequency of 0.1 Hz (average sleep duration or scrubbing interval up to 10 seconds), the system failure rate of a protected system is given by [161]:

$$Protected\ System\ Failure\ Rate = 0.5 \cdot \frac{R}{k} \cdot \lambda^2 n^2 \tag{3.5}$$

Using Equations 3.4 and 3.5, the system failure rate (SFR) can be estimated, Figure 3.32 shows how Hamming (63,57) code reduces the SFR under different soft error rates (SER) with the range between 10^{-12} to 10^{-7} (err/bit-hr), this range is based on findings from a previous study [152]. Supply voltage scaling increases the soft error rate due to reduction in critical charge [162]. Using the proposed technique, the reliability of the idle circuit can be improved to compensate for the reduction in reliability due to supply voltage scaling. This can be demonstrated through the following example. In the case study of Cortex-M0, the system failure rate (SFR) is plotted against bit error rate (BER) in Figure 3.32. It is assumed that a system operating at 1V supply and its bit error rate λ_{1V} is 10^{-7}. From Figure 3.32 it can be seen that for an unprotected system BER λ_{1V} of 10^{-7} corresponds to SFR f of 10^{-4}. A previous study has shown that BER increases by 3 times when supply voltage is reduced from 1V to 0.5V for a 65-nm technology node [153], which means the BER at 0.5V is $\lambda_{0.5V} = 3 \times 10^{-7}$. Using the proposed hardware correction through Hamming (63,57) code, it can be seen that even at BER $\lambda_{0.5V}$ of 3×10^{-7} the SFR f_p is significantly lower (10^{-8}) than the SFR f at 1V (10^{-4}) of an unprotected system. The impact of the proposed protection method on circuit idle power is also analysed in Section 3.3.4. It is shown in Figure 3.32 that Hamming code is effective in protecting the system when SER is between 10^{-12} to 10^{-7}; however it starts to lose effectiveness for bit error rate above 10^{-5}. It should be noted that the reliability of the stored parity bits is taken into account by the error detection (CRC) and correction (Hamming and software state recovery) and therefore they are protected.

Software state recovery is invoked when errors correction through Hamming fails due to more than one bit error in the same codeword. An experiment is conducted to analyse the effect of higher error rates (5 x 10^{-3} to 55 x 10^{-3}) on error detection capability of CRC-16 and error detection and correction capability of different Hamming codes. Two different error distribution coefficients were used (Equation 3.2). Figure 3.33a shows the case of evenly distributed errors using $\alpha = 0$ and Figure 3.33b shows the effect of clustered errors that are generated using $\alpha = 2$. From the two figures, it can be seen that the error detection capability of CRC-16 remains unaffected across higher error rates and different clustering coefficients, however error detection and correction capability of all Hamming codes reduce significantly as error rate increases all the more strongly when clustered errors are injected as shown in Figure 3.33b. It should be noted that conventional scan chain configuration is used in this work, where neighbouring flip-flops are connected together. The impact of clustered errors can be reduced by

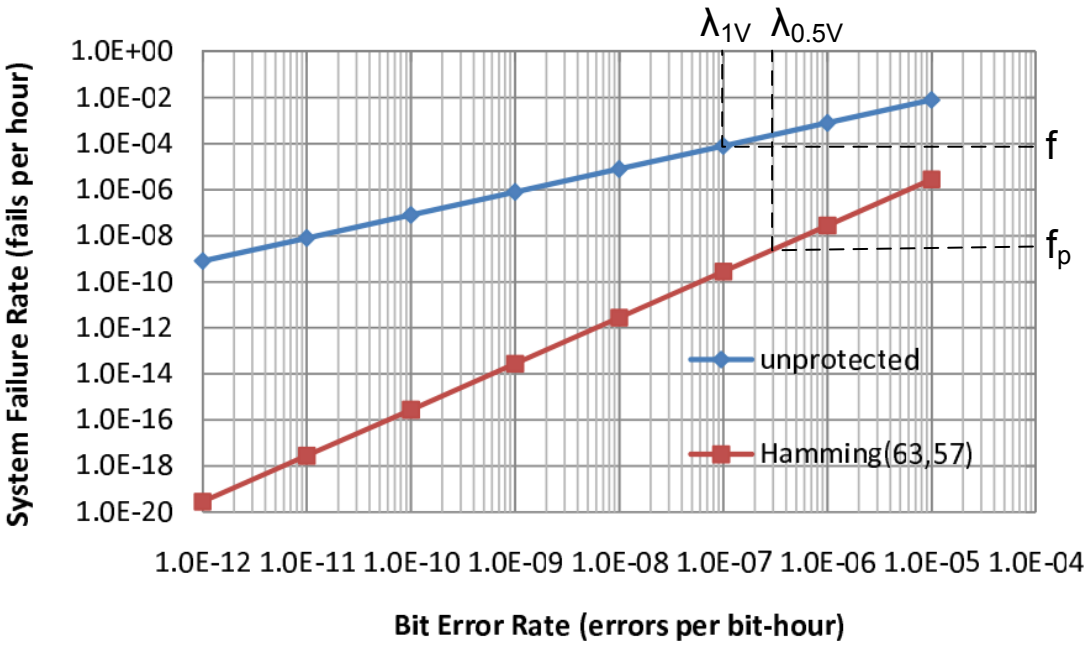
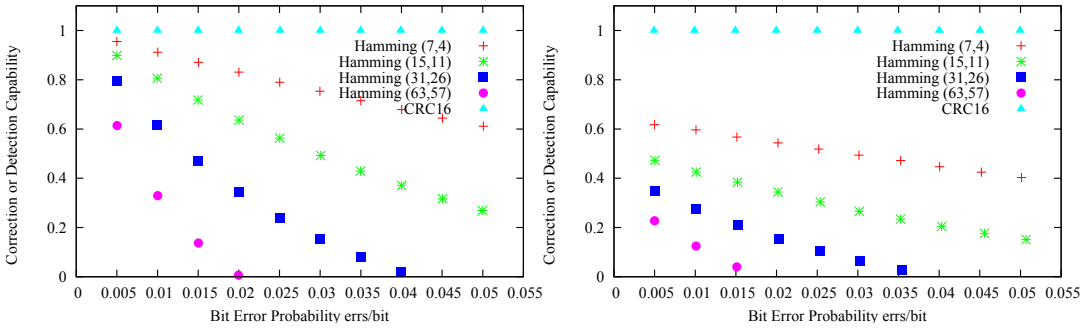


Figure 3.32: The system failure rate with and without hardware error recovery on Cortex-M0 as a case study



(a) Evenly distributed errors $\alpha = 0$

(b) Clustered errors $\alpha = 2$

Figure 3.33: Error correction capability of Hamming codes and error detection capability of CRC-16 at bit error probability of 0.005 to 0.05 errors/bit.

configuring scan chains to achieve maximum physical separation between consecutive flops. In this work clustered errors are dealt with using CRC detection and software state recovery. For software state recovery, the corrupted states are recovered through software checkpoints [163]. This is implemented by saving the architectural states before going to sleep mode. The states are placed in the ECC-protected SRAM for software state recovery, which is in "Always ON" power domain as shown in Figure 3.19. In the case of multiple errors in the same codeword, the software recovery is invoked that restores the saved states of the processor. Figure 3.31 shows that CRC-16 can detect 99.9978% of errors and subsequently all detected errors can be corrected by software state recovery. This clearly demonstrates the effectiveness of the proposed software state recovery method.

This work exploits the advantages of both techniques (hardware error correction and software state recovery) to achieve an effective state monitoring and recovery. For example, hardware error correction (Hamming code) is faster than the software state recovery technique. On the other hand, the software recovery technique has negligible area overhead and it can be used to recover burst errors, which cannot be corrected using Hamming code. It should be noted that, in the case of software recovery, the correction energy and latency cost is incurred only when errors are detected. The detailed trade-off analysis of the two techniques is discussed in the next section.

3.3.4 Trade-off Analysis

To protect a design with the proposed state monitoring and recovery technique, there is a trade-off between wake-up latency, area overhead and energy consumption. In this section, the trade-offs are analysed on the embedded processor shown in Figure 3.19. The effect of sleep rate on leakage power savings are examined. The deep sleep mode is not considered, because the whole circuit is completely powered down and there is no state integrity problem. The embedded system with the proposed state monitoring and recovery technique (Figure 3.19) is synthesised using a 65-nm STMicroelectronics gate library and Synopsys design compiler. Errors are injected in the system using the algorithm presented in Section 3.3.2 for evenly distributed errors ($\alpha = 0$) as well as for clustered errors ($\alpha = 2$).

The proposed scan-based state protection method incurs three costs for a design: firstly, it increases wake-up latency that is the time required for the circuit to switch from sleep mode to active mode; secondly, it incurs some area overhead due to the additional coding circuitry and parity storage; lastly, it consumes energy in every sleep and wake-up cycle. The generalised relation between coding scheme selection, scan chain configuration and the overhead is shown in this section.

For both Hamming and CRC protection, additional coding circuitry and parity storage lead to area overhead, which can be estimated using the following equation

$$A_{ov} = N_c \times A_c + N_s \times A_s + A_{PSMC} \tag{3.6}$$

where A_c is the area of a single coder, N_c is the number of coders used, A_s is the area of a single storage unit, N_s is the number of parity storage units, and A_{PSMC} is the area of the PSMC (Power and State Monitoring Control) block. The number of coders is proportional to the number of parallel scan chains. The number of parity storage units is proportional to the number of flip-flops in a design and the coding scheme used. Equation (3.6) can be rewritten as

$$A_{ov} = \frac{W_{sc}}{k} \times A_c + \frac{N_{ff} \cdot d}{k} \times A_s + A_{PSMC} \tag{3.7}$$

where W_{sc} is the number of parallel scan chains created, and N_{ff} is the number of flip-flops in a design. For any error correction code, a codeword consists of data bits for storing information and parity bits for protecting the information, k is the number of data bits and d is the number of parity bits in a codeword. It shows that the area overhead increases with the number of parallel scan chains W_{sc}, due to additional encoder and decoder. The area overhead also increases with the ratio of parity bit to data bit of the selected coding scheme due to redundancy storage (5th column in Table 3.6 and Table 3.7).

The scan chains need to be circulated to generate parity bits, and the number of clock cycles required are equal to the number of flip-flops in each scan chain L_{sc}. If the clock period is T_{clk}, the increase in wake-up latency can be calculated with

$$T_{ov} = T_{clk} \times L_{sc} \tag{3.8}$$

As expected, the hardware error detection and correction latency increases with the number of flip-flops in each scan chain. The power overhead for hardware error detection and correction has four components: scan shifting of circuit states P_{sc}, the operation of encoder and decoder P_c, parity storage power per bit P_s and the PSMC block P_{PSMC}

$$P_{ov} = P_{sc} + N_c \times P_c + N_s \times P_s + P_{PSMC} \tag{3.9}$$

$$= P_{sc} + \frac{W_{sc}}{k} \times P_c + \frac{N_{ff} \cdot d}{k} \times P_s + P_{PSMC} \tag{3.10}$$

The power overhead increases with the number of parallel scan chains because more coders are working in parallel. From Equations (3.8) and (3.10), energy consumed in each sleep and wake-up cycle can be calculated by

$$E_{ov} = P_{ov} \times T_{ov} \tag{3.11}$$

The above equation shows that the total energy is proportional to the number of flip-flops L_{sc} in each scan chain; this is further evaluated in the last column of Table 3.6 and Table 3.7. The above set of equations can be solved for any embedded processor to determine the overhead in terms of area, power, latency and energy. Table 3.5 shows the values of these parameters for hardware corrections (synthesised using 65-nm STMicroelectronics gate library) for all different Hamming codes used in this work. For example,

Hamming (63,57) has 57 data bits (k) and 6 parity bits (p). The area and power of the coder (A_c and P_c) reduces with each Hamming code; that is, from Hamming (63,57) to Hamming (7,4) due to reduction in parity generation tree. The rest of the variables are constant across all Hamming codes and their approximate values are shown in the last row of Table 3.5.

As discussed in Section 3.3.3, CRC-16 is chosen for error detection because of higher detection capability, low redundancy and simple encoder and decoder implementation. Table 3.6 shows the overhead on timing, area and energy consumption using four different scan chain configurations in comparison to a design without CRC unit. It can be seen that the area overhead increases linearly with the number of parallel scan chains ("Width"), from 3.5% to 6.1% as in the case of 16 scan chains to 80 scan chains respectively. This is because additional logic is required for encoding and decoding each additional scan chain through state monitoring and recovery block as shown in Figure 3.1. For all four scan chain configurations, the critical path length remains the same in comparison to a design without the CRC unit. The CRC unit does not increase the logic depth of the critical path because the system core is separated from the CRC unit (state monitoring block as shown in Figure 3.3(a)) through a clamp gate which is

Table 3.5: Parameters for calculating area, latency, power and energy consumption for hardware corrections

Code	k	d	$A_c(\mu m^2)$	P_c (μW/MHz)
Hamming (63,57)	57	6	665	0.47
Hamming (31,26)	26	5	282	0.25
Hamming (15,11)	11	4	119	0.15
Hamming (7,4)	4	3	36	0.06
$A_s \approx 19.5\ \mu m^2$/bit, $A_{PSMC} \approx 900\ \mu m^2$, N_{ff}=768 bits $P_s \approx 9.8\ nW$/MHz.bit, $P_{sc} \approx 25.8\ \mu$W per MHz, $P_{PSMC} \approx 0.3\ \mu$W per MHz				

k $\rightarrow$ Data bits, d $\rightarrow$ Parity bits, $A_c \rightarrow$ Coder Area, $P_c \rightarrow$ Coder Power

$P_s \rightarrow$ Storage Power, $A_{PSMC} \rightarrow$ PSMC Area, $P_{PSMC} \rightarrow$ PSMC Power

Table 3.6: Error detection (CRC-16) overhead on timing, area, latency, power and energy consumption using different scan chain configurations

Code	Scan Chain		Time	Cell Area		Latency	Power	Energy
	W	L	(ns)	(μm^2)	inc(%)	cycles	$\mu W/MHz$	(nJ)
Unprotected			11.9	37814				
CRC-16	16	49	11.9	39122	3.5	49	26.9	1.27
	32	25	11.9	39345	4.0	25	27.04	0.67
	57	14	11.9	39823	5.3	14	27.29	0.37
	80	10	11.9	40120	6.1	10	27.52	0.27

inactive during functional mode and adds negligible load capacitance on flip-flops connected to the state monitoring block. The next column ("Latency") shows the sleep and wake-up latency in number of clock cycles, which depends on scan chain length, longer scan chains require longer time to encode and decode the circuit's states. As expected the coding latency reduces from 49 to 10 at the cost of area overhead, which increases from 3.5% to 6.1%. The next column shows the power consumption of the CRC unit, which is estimated by Synopsys PrimeTime PX. The power consumed by the CRC unit increases with the number of scan chains due to additional gates in state monitoring blocks to encode more scan chains in parallel. The last column shows the overall energy consumed by the CRC protected system. It can be seen that the energy consumption reduces from 1.27-nJ to 0.27-nJ with reduction in scan chain length from 49 to 10. This is because energy consumption reduces with latency (number of clock cycles), which is dependent on scan chain length, clearly showing a trade-off between energy consumption and area overhead across different scan chain configurations.

Next, the implementation trade-off is analysed using four different Hamming codes where each is implemented with different scan chain configurations. The results are shown in Table 3.7. The critical path length is shown in the third column, which stays the same for all configurations with or without Hamming code. This confirms that the state monitoring does not affect the timing of the design. The next column shows the sleep and wake-up latency in the number of clock cycles; as expected the coding latency varies with the scan chain length from 7 to 196 as in the case of Hamming (63,57) and Hamming (7,4). Area overhead is shown in the next column; it can be seen that Hamming (63,57) has the smallest area overhead (8.6%) in comparison to a design without error correction capability. Hamming code requires redundancy for code storage, which increases from Hamming (63,57) to Hamming (7,4) resulting in higher area overhead (up to 30.9%). The power consumption of the ECC unit is shown next, which is lowest (27.69 μW per MHz) in the case of Hamming (63,57) code, but increases with different Hamming codes due to additional load capacitance of redundant storage elements; that is, when the redundancy increases from 8.6% to 30.9% as in the case of Hamming (63,57) to Hamming (7,4). The last column shows the energy consumption; as expected this varies with the scan chain length resulting in lower latency and energy consumed for encoding and decoding. It can be seen that the lowest and the highest energy consumption are 0.2-nJ and 6.09-nJ, for scan chain lengths of 7 and 196 respectively.

The results presented in Table 3.6 and Table 3.7 demonstrate the trade-off between energy consumption, latency and area overhead across different scan chain configurations. Increasing the number of scan chains increases the area overhead and coding power; however the wake-up latency and energy consumption is reduced. From the two tables it can be observed that a scan chain configuration of 57x14 gives the best results. Area overhead is 8.6% as in the case of Hamming (63,57) (Table 3.7) and 5.3% as in the case of CRC-16 (Table 3.6). Since this configuration requires only 14 clock cycles for

Table 3.7: Error correction (Hamming) overhead on timing, area and energy consumption using different scan chains configuration

Code	Scan Chain		Time	Cell Area		Latency	Power	Energy
	W	L	(ns)	μm^2	inc(%)	cycles	$\mu W/MHz$	(nJ)
Unprotected			11.9	37814				
Hamming (63,57)	57	14	11.9	41070	8.6	14	27.69	0.39
	114	7	11.9	41734	10.4	7	28.46	0.2
Hamming (31,26)	26	31	11.9	42861	13.4	31	28.15	0.86
	52	16	11.9	43145	14.1	16	28.46	0.46
	104	8	11.9	43997	16.4	8	29.38	0.24
Hamming (15,11)	11	72	11.9	43829	15.9	72	28.92	2.10
	22	36	11.9	43948	16.2	36	29.53	1.03
	44	18	11.9	44305	17.2	18	29.84	0.54
Hamming (7,4)	4	196	11.9	49357	30.5	196	31.07	6.09
	8	98	11.9	49393	30.6	98	31.38	3.04
	16	49	11.9	49501	30.9	49	32.15	1.57

state monitoring and recovery, the energy overhead is 0.39-nJ and 0.37-nJ in the case of Hamming (63,57) and CRC-16 respectively.

Next the overhead of software state recovery is discussed and compared with Hamming (63,57) code when using the same scan chain configuration (57x14). In terms of area it has 5.3% overhead due to using CRC-16 for error detection and the state recovery power is 13.69-μW per MHz (calculated using Synopsys PrimeTime PX). The software state recovery has high wake-up latency of 184 clock cycles and energy overhead of 2.67-nJ. In comparison to Hamming (63,57), software state recovery introduces 13x higher wake-up latency and consumes about 7x more energy.

Error correction has encoding and decoding energy overheads in each sleep cycle and leakage power overhead due to the additional circuitry for coder and code storage, adding to the idle power of the design. Figure 3.34 shows the trade-off between idle power and sleep frequency (number of sleep and wake-up cycles per second) of the processor core with and without protection, taking into account all sources of leakage in the design, including the storage of parity bits during sleep mode. For illustration purposes, 20% duty cycles (activity factors) are used, and activity factor is calculated by taking the ratio of total active time to total runtime of the processor. The leakage power of the processor core in sleep mode is estimated by using Synopsys PrimeTime PX. Figure 3.34 shows the relation between idle power and sleep frequency for retention voltage V_{ret} of 1V and 0.5V with and without protection. Four different scenarios are created: $V_{ret} = 1V$ without protection, $V_{ret} = 0.5V$ without protection, $V_{ret} = 0.5V$ with hardware error correction and $V_{ret} = 0.5V$ with software state recovery. The idle

power was normalised to that of the unprotected system with $V_{ret} = 1V$. It can be observed that the idle power is lowest for an unprotected system with retention voltage $V_{ret} = 0.5V$ when compared with hardware error correction and software state recovery. This is because hardware error correction and software state recovery require additional circuitry to improve system reliability. When comparing the two recovery methods, software recovery has higher idle power, which is due to its higher latency (number of clock cycles for state recovery) in comparison to hardware recovery method. For example, at sleep frequency of 100 Hz, Figure 3.34 shows that hardware error correction has 0.21 idle power while software recovery has 0.4 idle power. For these two error recovery methods, idle power is also related to sleep frequencies. Idle power is similar with or without protection until sleep frequency is greater than 10-Hz; this is because when number of sleep and wake-up cycles is small, the energy used for error checking is negligible. For software state recovery, when the sleep frequency is higher than 400-Hz, its idle power is greater than the unprotected system with $V_{ret} = 1V$ (without voltage scaling), which means it consumes more energy than it saved.

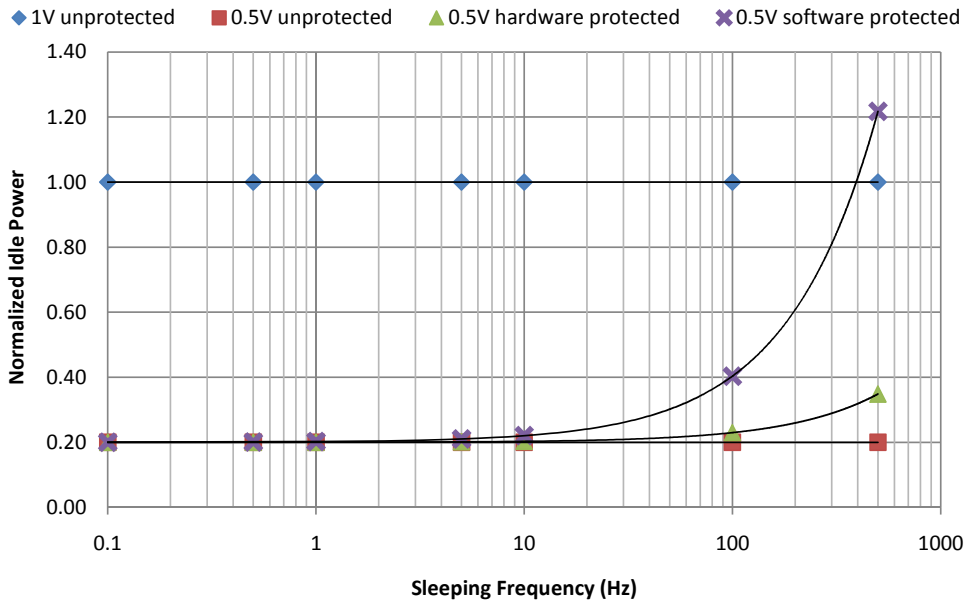


Figure 3.34: Trade-off analysis between leakage power saving and sleep frequency of the processor core at 0.5-V supply voltage

3.4 Concluding Remarks

In this chapter an efficient design method is proposed to improve the reliability of state retention design by protecting state retention registers through monitoring and recovery. This is achieved by exploiting the available scan chain infrastructure without affecting manufacturing test and critical paths of state retention designs. Using synthesised designs and various scan chain configurations, it is shown that the proposed method can be incorporated into the state retention design flow. To study the effectiveness of the proposed method, it is first used to improve the reliability of FIFO design implemented

in FPGA and shows 100% error detection both for single error and burst error injection. In addition, it achieves 100% error correction in the case of single errors. The proposed method is further validated using a commercial microprocessor ARM Cortex-M0 implemented in FPGA, and additional software recovery is used together with hardware detection for burst error correction. Under low bit error rate (10^{-5} to 10^{-12}) hardware error correction can improve system error rate by at least 2 order of magnitude, under high error probability (0.5 to 10^{-5}) hardware error detection can detect 99.6% errors and software recovery can be used. Finally both FIFO and microprocessor design were synthesised using ST 65nm standard cell library to analyse the implementation trade-offs of the proposed method. In the case of FIFO design, the area overhead of hardware error detection is between 2.8% and 9.2% depends on scan chain configurations and hardware error detection and correction using Hamming (63,57) is 15.9%. In the case of the microprocessor design, the area overhead of hardware error detection is between 3.5% and 6.1% depends on scan chain configurations and hardware error detection and correction using Hamming (63,57) is 8.6%.

Chapter 4

Improving the State Integrity of Flip-flops under PVT Variation

In Chapter 1 Section 1.2.2, it was shown that power gating [37] and supply voltage scaling [41] are two state retention design techniques for effective standby circuit leakage power reduction. In Chapter 2 Section 2.3, it was shown that process variation significantly impacts critical charge of storage elements, supply voltage scaling further exacerbating the problem. In Chapter 3 a method was proposed for improving reliability of state retention designs, which was validated using two case studies. An important objective of the research is to investigate the impact of process, voltage and temperature (PVT) variations on the reliability of state retention registers in low power designs. The research aim of this chapter is to maximise the leakage saving using supply voltage scaling while improving the reliability of flip-flops under PVT variation. Monte-Carlo simulations and measured results on 82 dies are used to demonstrate flip-flop state integrity sensitivity due to PVT variation. Binary search based Minimum Retention Voltage (MRV) characterisation is proposed to determine the MRV of individual die in the presence of PVT variation for aggressive leakage power reduction. A control flow is proposed for state monitoring and protection of flip-flops, which uses horizontal and vertical parity checks for multi-bit error detection and single bit error correction. Silicon results show that state integrity is preserved, while reducing leakage power during standby mode.

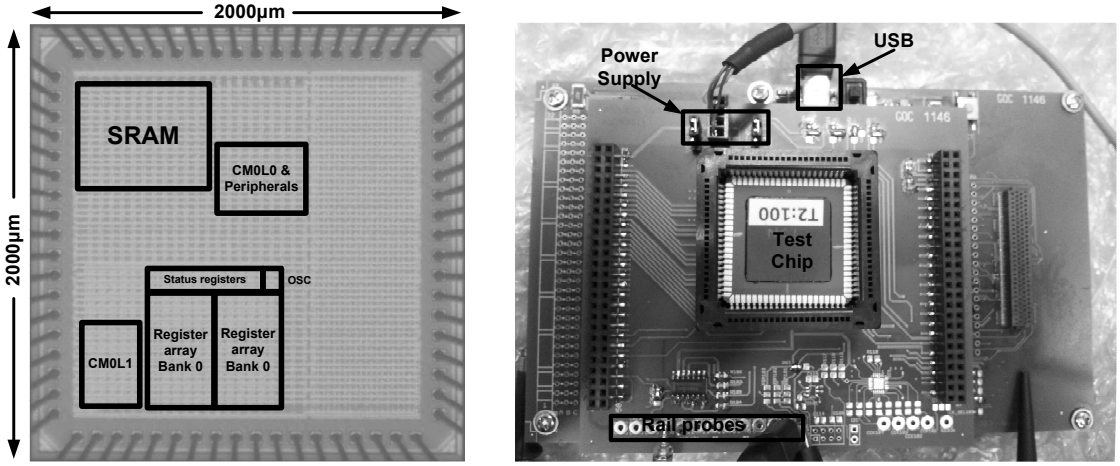
This chapter is organised as follows: In Section 4.1, Monte-Carlo simulations and test chip measurement results are used to demonstrate that state integrity of flip-flops are sensitive to PVT variations. The new MRV characterisation and flip-flop state-protection technique are presented in Section 4.2. The test chip implementation is presented in Section 4.3. Experimental results to demonstrate leakage power saving with state retention integrity are presented in Section 4.4. Conclusions on the approach are drawn in Section 4.5.

4.1 State Integrity Challenges under PVT Variation

Supply voltage scaling is an effective technique for reducing standby mode leakage power, and is frequently used in energy-constrained designs [38]. Recent research has shown that it effectively reduces sub-threshold and gate-leakage power in deep-submicron designs [38, 77]. This is because of the negative exponential relationship of leakage power and supply voltage, when $V_{gs} \approx 0$ [164]. Power minimization through supply voltage scaling during sleep state was proposed [41, 76]. Maximum leakage power saving is achieved under the lowest supply voltage where flip-flops can still retain their states; this voltage is referred to as Minimum Retention Voltage (MRV).

State integrity is referred to as the capability of flip-flops to retain their logic value last clocked-in. To analyse state integrity of voltage scaled state retention flip-flops, two identical register arrays of 8192 flip-flops referred to as retention register block is implemented in TSMC 65nm "LP" low leakage technology with nominal operating voltage of 1.2V using Unified Power Format (UPF) design flow refer to latest IEEE 1801-2013 (UPF 2.1) Standard [165] and standard EDA tools (Synopsys Inc, Mentor Graphics Inc). The test chip is shown in Figure 4.1, where Figure 4.1-(a) shows the die photo of the test chip with detailed implementation presented in Section 4.3, and Figure 4.1-(b) shows the test board photo. Measurements presented in this work are based on 82 test chips. As shown in Figure 4.1-(a), the retention register blocks are located on the bottom left-hand side of the die, and the parity storage is placed above the register block. A Cortex-M0 micro-controller from ARM Ltd (referred to as CM0) is used in this work for state monitoring and it can be seen on the left-hand side of the register block. The detail description of CM0 is in Appendix A A pair of oscillators has also been used to measure delay variation due to process, voltage and temperature variations, when considering inter-die and intra-die process variation. The oscillators are located next to the parity storage unit. Due to the small size, these are not marked explicitly on the die photo. The layout is part of a 2x2-mm system on chip (SoC), and the rest of the SoC is made up of SRAM for instruction and data storage. The test board is shown in Figure 4.1-(b), which provides voltage rail probe-points, power supply connections and USB interface to communicate with the host computer through an ASCII debug protocol.

In this work, the First Failure Voltage (FFV) of a flip-flop is defined, such that scaling down supply voltage to FFV leads to the first bit(s) failure in a design consisting of n flip-flops, where bit-failure refers to the change in stored logic value from the initial (or correct) value. Note, that single or multiple bits failures are possible at FFV. Due to process, voltage and temperature variation, the FFV of a given design varies from die to die. To ensure state integrity, it is important to analyse this change in FFV of state-retention flip-flops. Using measured results from 82 dies, this section analyses the change in FFV due to process, voltage and temperature variation. Section 4.1.1 shows FFV distribution from 82 dies. Section 4.1.2 analyses change in FFV due to within die



(a) Die photo (b) Test board

Figure 4.1: Test silicon fabricated and packaged for evaluation.

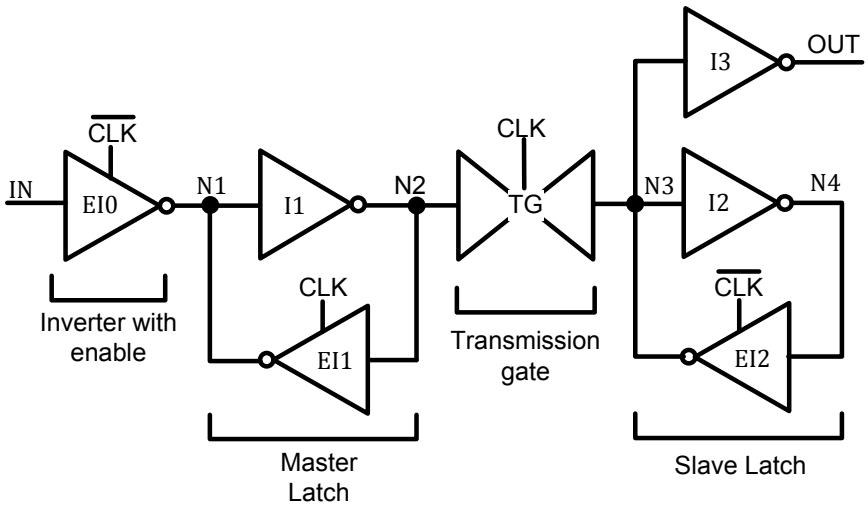
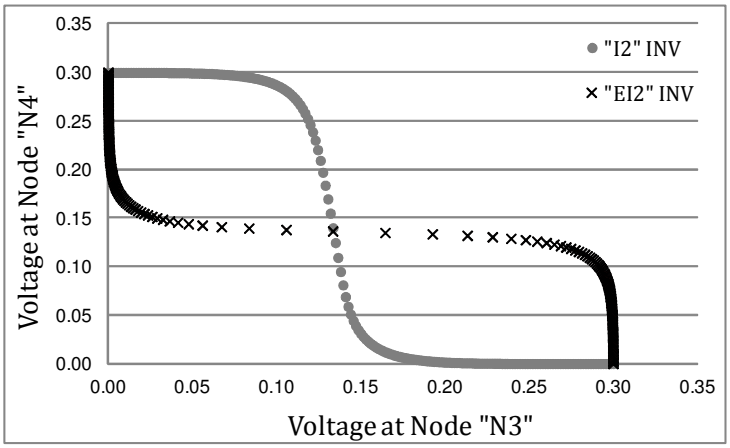


Figure 4.2: Schematic of master-slave flip-flop.

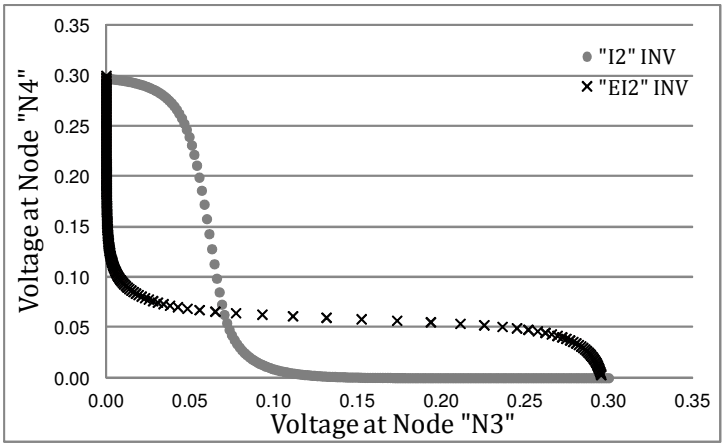
process and voltage variation, and finally Section 4.1.3 analyses change in FFV due to temperature variation.

Figure 4.2 shows the schematic of master-slave flip-flop commonly used in modern digital designs. The master latch is transparent when the clock is low and the slave latch is transparent when the clock is high. The slave latch that is made of two cross-coupled inverters is used for state retention at low supply voltage. In theory, the latch is capable of retaining its state at a very low supply voltage, given that the design is not affected by process variation; that is both PMOS and NMOS of the two cross-coupled inverters (I2 and EI2) have the same drive strength; and there is no noise for example, due to supply voltage fluctuation and radiation-induced soft error. This is because transistor ON-current is always higher than its OFF-current.

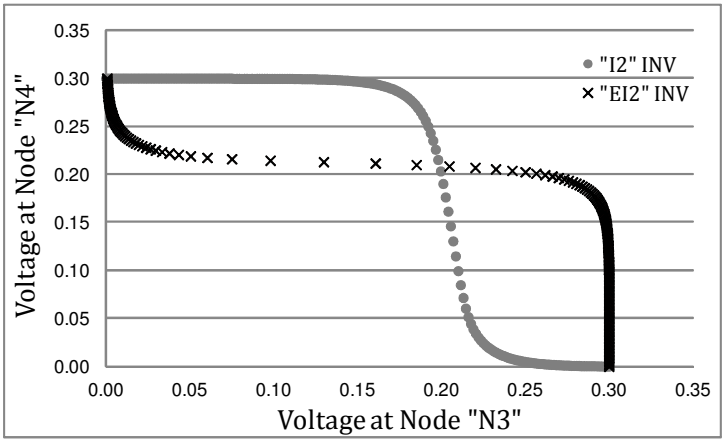
In real-life digital circuit designs, however, there are environmental noises and PVT variation. It was shown in Section 2.4 that process variation reduces the critical charge



(a) Typical Process at 0.3-V



(b) 3σ Fast-Slow process corner at 0.3-V



(c) 3σ Slow-Fast process corner at 0.3-V

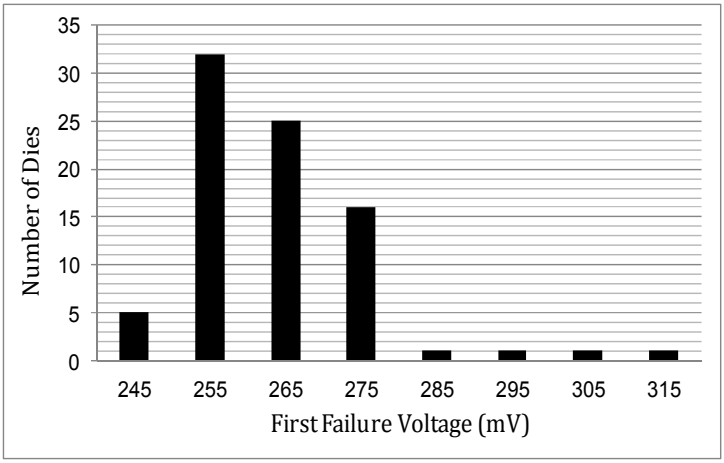
Figure 4.3: Simulated results showing noise margins of a typical flip-flop for state retention (Figure 4.2) when operating at 0.3-V, and reduced noise margins due to process variation at fast-slow and slow-fast corners.

of flip-flops makes them more sensible to soft errors. PVT variation also reduces the FFV of flip-flops. To get an insight into FFV spread of state-retention flip-flops, HSPICE was used to simulate the effect of process variation on state-retention capability of a flip-flop. The simulation was carried out on the slave latch (Figure 4.2), using design parameters from typical, fast-slow and slow-fast process corners of TSMC 65-nm Low-Power design library. The results are shown in Figure 4.3. In all three plots, the voltage transfer curve of inverter 'I2' is represented by circular dots, and that of 'EI2' is represented by black crosses. X-axes show voltage at node 'N3' and y-axes show voltage at node 'N4' (output, Figure 4.2). Figure 4.3-(a) shows a typical design operating at 0.3-V. It can be seen that the logic threshold voltages of both inverters are equal at about $0.5 * V_{dd}$, leading to symmetric noise margins for storing both logic values in the slave latch. However, when considering a fast-slow process corner, operating at 0.3-V (Figure 4.3-(b)), the logic threshold voltages of both inverters reduce to about $0.23 * V_{dd}$, leading to asymmetric noise margins for storing logic-0 on both nodes 'N3' and 'N4'. This means that a small noise can convert logic-0 to logic-1 on 'N3' and 'N4', leading to data corruption of stored states at low-supply voltage. Similarly, when considering a slow-fast process corner, operating at 0.3-V (Figure 4.3-(c)), the logic threshold voltages of both inverters increase to about $0.7 * V_{dd}$, again leading to asymmetric noise margins for storing logic-1 on both nodes 'N3' and 'N4', and a small noise can convert logic-1 to logic-0 on both nodes, leading to state corruption at low-supply voltage. These results clearly demonstrate that the state-retention capability of a voltage-scaled flip-flop is affected by process variation, and simulated results (Figure 4.3) reveal that due to process variation, the noise margin of a flip-flop gets skewed leading to variation in FFV (First Failure Voltage).

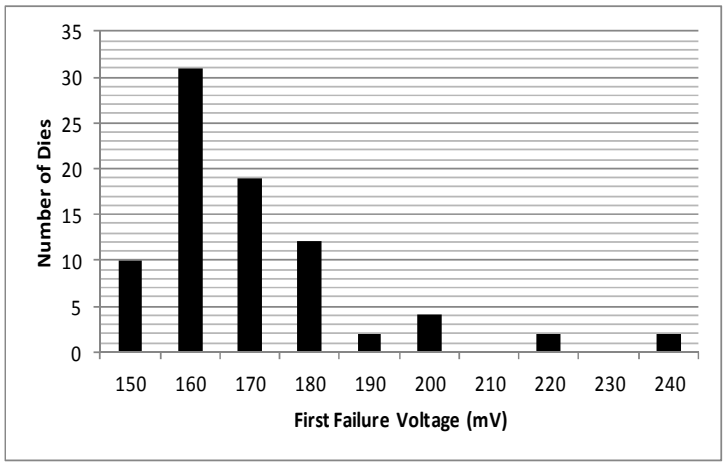
4.1.1 Measuring Inter-die Process Variation Impact on State Retention

Measurements were taken for 82 packaged dies and the FFV results are shown plotted in Figure 4.4-(a). The spread in FFV is clearly visible. This measurement was carried out at room temperature (25°C) using 82 dies, each with 8192 flip-flops, and with the implementation set-up shown in Figure 4.1. For this measurement, a test board was connected with a host computer through USB interface, and a control program written in Python script was used to communicate between the host computer and the test board. The FFV is found using a binary search algorithm with resolution of 1-mV per iteration, starting from 400-mV, until first bit failure is observed (Figure 4.11). Each iteration consisted of the following five steps:

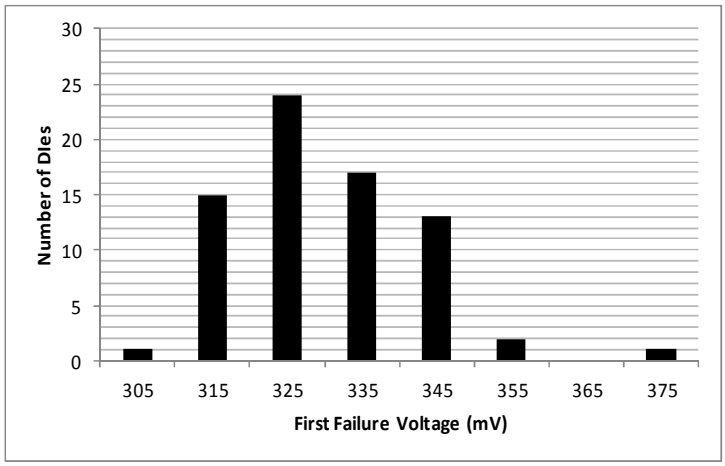
1. Voltage of the design was set to 1.2-V.
2. A single logic value (logic-0 or logic-1) was stored in all 8192 flip-flops, referred to as *initial logic state*.



(a) 65nm Test-chip Measured Results

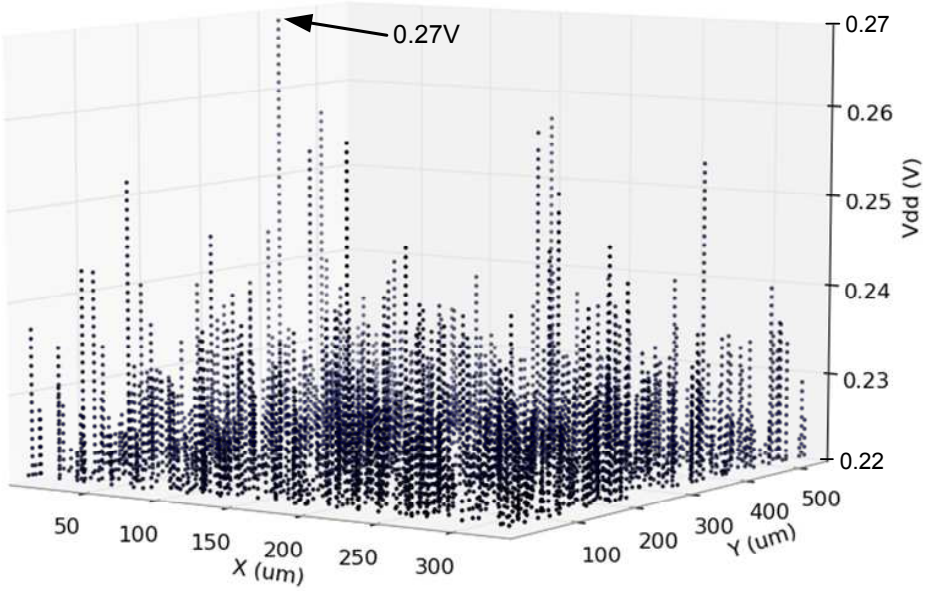


(b) Monte-Carlo Simulation for 65nm

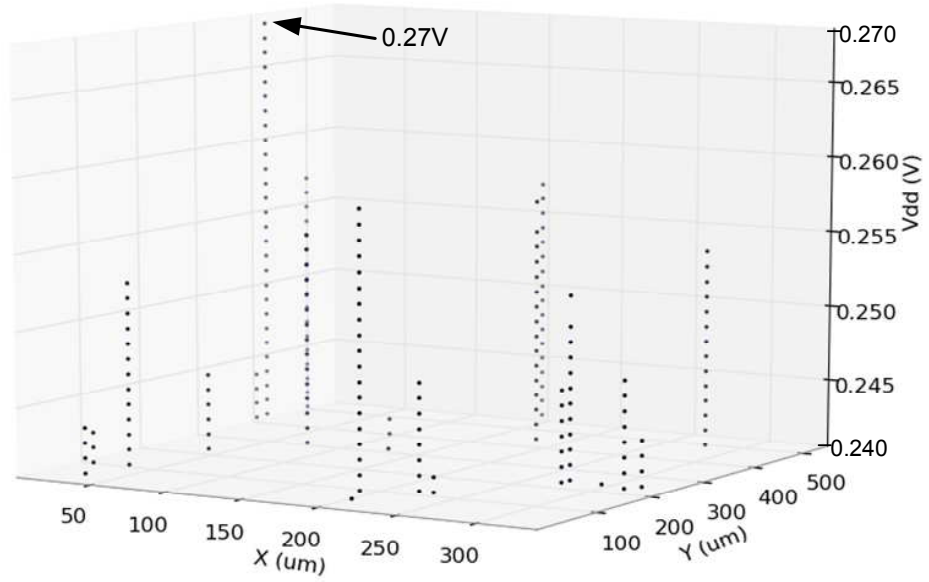


(c) Monte-Carlo Simulation for 45nm

Figure 4.4: Results from 82 dies and Monte-Carlo simulations showing the spread of first failure voltage point of voltage-scaled flip-flop for state retention in room temperature (25°C).



(a) failure voltage map from 220-270 mV
of retention register block



(b) failure voltage map from 240-270 mV
of retention register block

Figure 4.5: Measured results showing failure bit locations mapped on the circuit physical layout in the retention register block of the test chip.

3. Supply voltage was reduced to a lower voltage with a fall-time of 40-μs and this was held for 10-sec.

4. Supply voltage was raised back to 1.2-V with a rise-time of 40-μs.

5. The flip-flop state values were read out and compared with the initial logic state to determine if FFV has been observed.

Each iteration was executed 10-times to minimise the effect of jitter and most common value was recorded. The maximum jitter of 2-mV was observed when charge time was 40-μs and this value increased to 18-mV with a shorter charging time ($\leq$ 4-μs). From Figure 4.4-(a), it can be seen that the First Failure Voltage (FFV) point of each design varies from die to die, and for these 82 dies FFV is in between 245-mV to 315-mV, with 95% dies exhibiting their FFV below 285-mV. The measured results shown in this work use Standard Vth (SVT) cell. Simulation results using Low Vth (LVT) and High Vth (HVT) cells show that LVT cell has the lowest FFV and HVT cell has the highest FFV. Figure 4.4-(b) shows the Monte-Carlo simulation result for the same 65-nm TSMC technology library. The HSPICE script used to generate this result is described in Appendix D. This is used to analyse how well simulated results correlate with measured results (Figure 4.4-(a)). The effect of process variation is incorporated by varying three parameters, which include: gate length (L), threshold voltage (V_{th}), and mobility (μ_{eff}) (Mobility varies due to variation in effective strain in a strained silicon process [90]). These parameters follow Gaussian distribution ($\pm 3\sigma$ variation) with standard deviations of 4% for L, 5% for V_{th} and 21% for μ_{eff}. It can be seen that the overall distribution trend remains the same while the mean FFV has shifted to lower voltage. This is because simulation results do not take into account environmental noise and inductive effects. It can also be observed that the spread of FFV is slightly wider in simulation than measured results; this is because the effect of process variation on fabricated devices is less than simulated results. Similarly, the FFV for a 45-nm technology library [166] was simulated. Results are shown in Figure 4.4-(c). When comparing it with simulated results of 65-nm technology library (Figure 4.4-(b)), it can be observed that the overall distribution trend remains the same, however the mean FFV has shifted to a higher voltage due to higher process variation.

4.1.2 Effect of within Die Process and Voltage Variation

To analyse failure voltage across 8192 flip-flops within a single die, a die exhibiting nominal process characteristic is used and measurement set-up outlined in Section 4.1.1. Figure 4.5 shows measured results from the test chip to demonstrate the failure voltage behaviour of 8192 flip-flops and their individual XY co-ordinates within the design layout (Figure 4.1-(a)). Figure 4.5 shows the location of failed flip-flops as observed on the test chip. The 'X' and 'Y' axes show physical location of each flip-flop and indicate the

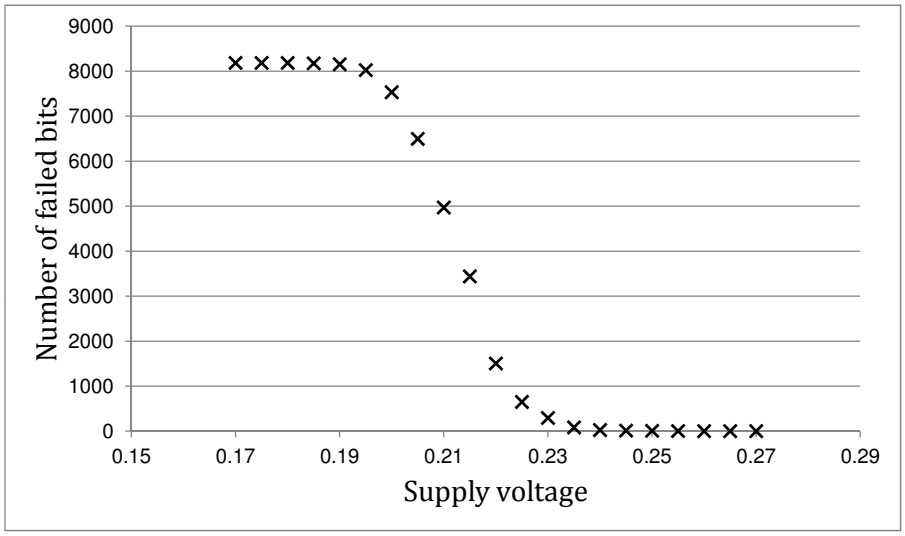


Figure 4.6: Measured results showing the distribution of failing voltage point of flip-flops at reduced supply voltage.

distance (in μm) from the bottom left corner of the retention register block (Figure 4.1). The Z-axis shows the supply voltage during retention mode. It can be observed that the First Failure Voltage (FFV) occurs at 270-mV, and this flip-flop continues to fail with further reduction in supply voltage. A few subsequent failure points are at about 260-mV. In general, over all flip-flops, when the supply voltage is $\leq$ 240-mV, increasing numbers of flip-flops start to fail, and this is shown in Figure 4.6 using 5-mV step size. For this measurement (Figure 4.6), ten test runs were conducted, and the plot shows the average number of failed bits over all test runs. Figure 4.6 shows that the first bit failure is observed at 270-mV, and the number of failed bits increase with further reduction in supply voltage until the supply voltage is reduced to 190-mV, where all flip-flops failed to retain initially stored logic values.

To get an insight into failure pattern across all 82 dies, a measurement is taken at room temperature (25°C) to determine the FFV of each die, and voltage difference between the first and subsequent failing flip-flops. The results are shown in Figure 4.7, where X-axis show FFV of each die, and Y-axis show the voltage difference between the first and subsequent failing flip-flop for each die. For example, in the case of Die-3, FFV is observed at about 250-mV, and the voltage difference (Y-axis) is 0, representing multi-bit failure (two or more flip-flops) at FFV. Similarly, in the case of Die-2, FFV is observed at about 270-mV, but the difference between the first bit failure and subsequent bit failure is about 9-mV. As can be seen, the voltage difference between first and subsequent flip-flop failures is highest (66-mV) in the case of Die-1. In general, when using a discrete step size of 5-mV, across all dies it was found that 20.73% of all dies show multiple bit failure at First Failure Voltage (FFV) point and the rest (79.27%) show only single bit failure at FFV. This finding is exploited in our proposed technique (Section 4.2), which

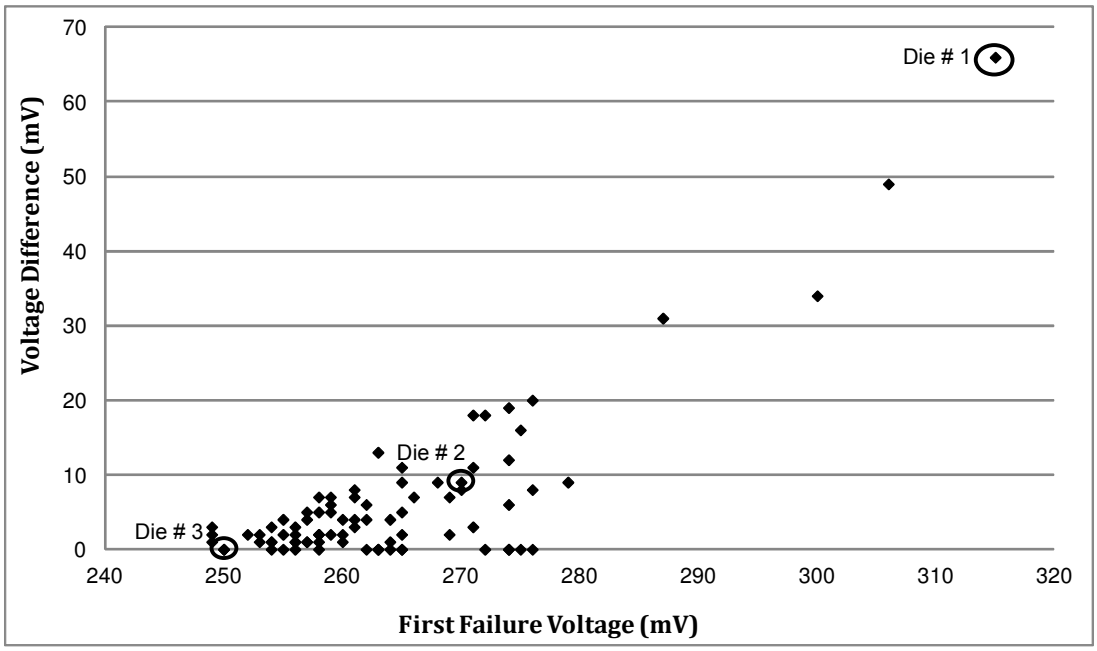


Figure 4.7: Measured results from 82 dies showing voltage difference between flip-flop's first failure voltage and subsequent failure voltages, three dies were selected from the population and re-numbered 1-2-3.

employs a simple parity-based error detection and correction technique for multi-bit error detection and single bit error correction.

4.1.3 Effect of Temperature Variation

The effect of temperature variation on state retention voltage of a flip-flop was also examined for three dies marked in Figure 4.7. These three dies represent both nominal and corner cases. First Failure Voltage (FFV) on the following four temperatures are measured: 25°C, 41°C, 56°C, and 79°C. The temperature of the test chip was raised using a temperature chamber. Figure 4.8 shows the relationship between FFV and temperature. For all three dies, as expected, it was found that FFV increases with temperature. This is because transistor leakage current increases with temperature, while drive current (I_{on}) decreases [164]. In the case of the master-slave flip-flop shown in Figure 4.2, the state integrity of a storage node (N3 or N4) depends on the charge stored and the feedback current. As temperature increases, this feedback current reduces due to increase in leakage current and reduction in drive current, which negatively affects state retention capability of storage node at higher temperatures. This means that the state retention voltage of a flip-flop has to be raised at higher temperatures to ensure state integrity.

To get an insight into the combined effect of process, voltage, and temperature variation on a given design. Using Die-2 (Figure 4.7), within-die delay variation is measured by changing the supply voltage and temperature, using two identical ring oscillator chains

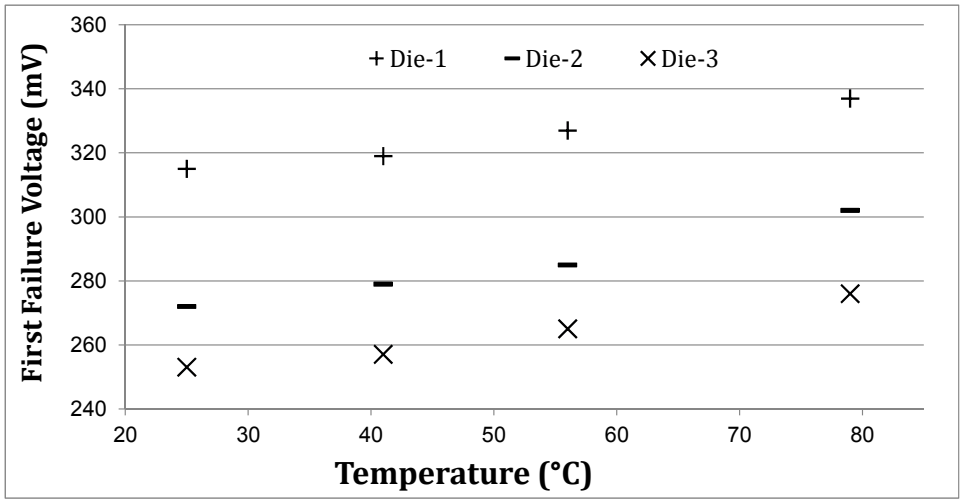


Figure 4.8: Measured results showing the first failing voltage point of flip-flops due to within-die temperature variation.

(OSC) each with 95 NAND gates. The results are shown plotted in Figure 4.9, where X-axis show supply voltage and Y-axis show normalised delay variation at four temperature points. Normalised delay variation is calculated by taking the relative mean difference of measured delay between the pair of OSC, at each temperature and supply voltage point, and it is normalised with that of 1.2-V supply voltage at 25°C temperature. It can be seen that normalised delay variation is smallest at nominal supply (1.2-V) and room temperature (25°C). It increases by up to 15x at 79°C when supply voltage is reduced from 1.2-V to 0.5-V. This shows that at lower voltage and higher temperature, the effect of V_{th} variation has greater impact at oscillator frequency variation [167]. This means that due to process variation, the state integrity of a flip-flop (Figure 4.2) is more vulnerable at reduced supply voltage and higher temperature. Note that these results (Figure 4.8 and Figure 4.9) are specific to this technology library and are shown for illustration purposes only. For smaller geometry (below 65-nm), temperature spread and delay variation may be more strongly affected by, for example, additional mobility caused by increased mechanical strain process engineering and reduced threshold voltage.

As shown in Figure 4.8, First Failure Voltage (FFV) point of a flip-flop increases with increase in temperature. This means "Sleep State" voltage should take temperature variation into account to ensure state integrity. This has an effect on leakage power consumption of a design in "Sleep State". To get an insight into voltage scaling and leakage power, Figure 4.10 shows "Sleep State" leakage power by measuring I_{ds}, and varying the supply voltage after setting $V_{gs} = 0$. The measurements were carried out on a test chip (Die-2, Figure 4.7) under four different temperature settings: 25°C, 41°C, 56°C, and 79°C. The x-axis shows the supply voltage ranging from 0.3-V to 1.2-V. The y-axis shows the normalised leakage power plotted on a log-scale. It can be observed that leakage power reduces exponentially with reduction in supply voltage, and 97.5% leakage power minimization is possible by reducing the supply voltage from 1.2-V to

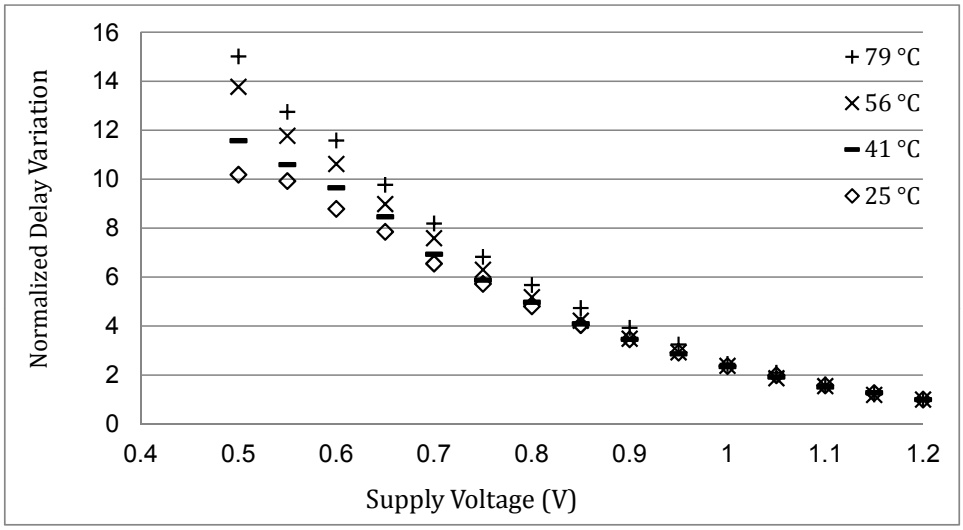


Figure 4.9: Measured results of Intra-Die PVT Variation on delay of the test chip. These results demonstrate that due to change in temperature the effect of within die process variation gets worse as shown by within die higher normalised delay variation.

0.3-V. The effect of temperature variation can also be observed: as can be seen, at a given voltage, leakage power increases with temperature. The leakage power at 79°C is an order of magnitude higher than at room temperature (25°C).

In this work, Minimum Retention Voltage (MRV) is defined as the scaled supply voltage value, at which all flip-flops in a given design can still preserve their state integrity. For a given technology and flip-flop design, the minimum retention voltage of a design has to be characterised across all process and temperature corners to ensure state integrity. From the trend shown in Figure 4.7 (Section 4.1.2) and Figure 4.8 (Section 4.1.3), an important observation can be made. Due to process variation, MRV varies from die to die and characterizing each die separately will not only ensure state integrity but can also minimise retention voltage per die, thus reducing leakage power. In this case, MRV can be calculated by adding a voltage margin (referred to as Retention Voltage Margin (RVM)) to First Failure Voltage (FFV) of the given die, that is $MRV = FFV + RVM$. For example setting RVM to 54-mV for all dies (see Section 4.2.1 for details of calculating RVM), the MRV of Die-1 is $315 + 54 = 369$-mV, and that of Die-3 is $249 + 54 = 303$-mV. Therefore setting MRV for each individual die separately is beneficial to leakage power minimization, when compared to a technique that sets the MRV of all dies using worst-case process and temperature corners. This observation is exploited in the proposed technique (Section 4.2), which employs a characterisation algorithm to identify the MRV of each die to minimise leakage power.

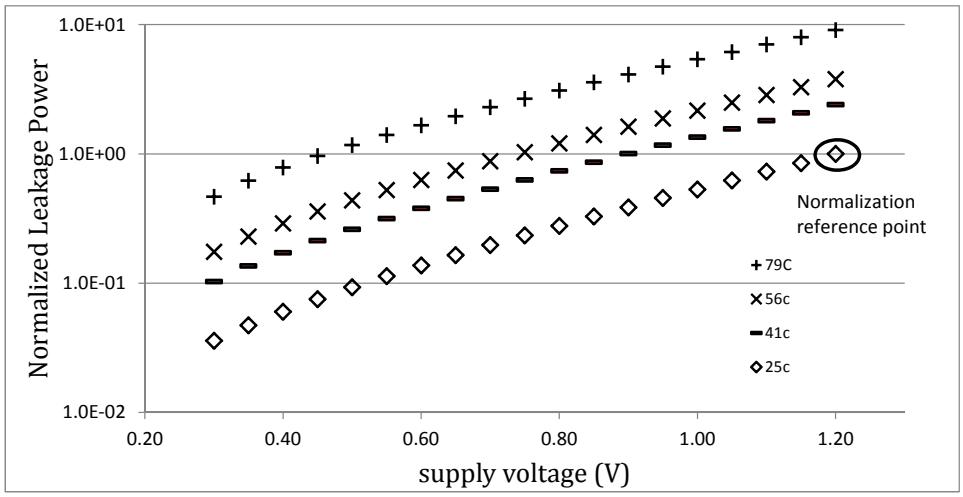


Figure 4.10: Measured test chip leakage power normalised to 1.2V nominal supply voltage at 25°C.

4.2 PVT Aware State Protection Technique

PVT variation analysis discussed in the previous section shows two important observations. Firstly, 79% of all dies exhibit single bit failure at FFV (when using 5-mV discrete voltage steps), while the rest show multi-bit failure. Secondly, MRV characterisation per die is beneficial for leakage power minimization. These two observations are used to develop a simple and effective technique to improve state-integrity of voltage scaled flip-flops under process, voltage and temperature variation. The proposed technique consists of the following two parts:

1. Section 4.2.1 presents a characterisation algorithm that is used to determine the MRV of a given die; this is because the MRV of each die varies due to process variation as observed in Figure 4.7. The characterisation is an offline process and is performed before the die is used.

2. Section 4.2.2 presents a control flow for error detection and single-bit error correction, which relies on horizontal and vertical parity; The control flow is an online process which monitors the flip-flops states. If any errors are detected, it raises the pre-characterised MRV to reduce subsequent error possibility. This is because the MRV can change with the operating environment such as rising temperature. The prototype of the proposed control flow is implemented in the host computer using Python script, which provides voltage scaling by controlling an external power supply to the test chip (Figure 4.1-(b)).

Input: Initial Retention Voltage (IRV), Voltage Scaling Resolution (VSR), Retention Voltage Margin (RVM)
Output: Minimum Retention Voltage (MRV)
1: $V_{correct}$ = IRV; $V_{fail} = 0$
// $V_{correct}$ *is the lower bound of supply voltage for correct state retention and* V_{fail} *is the upper bound of supply voltage for failed state retention*
2: Current Supply Voltage = $\frac{V_{correct}+V_{fail}}{2}$
3: **while** $V_{correct} - V_{fail}$ >VSR **do**
4: **if** error detected **then**
5: V_{fail} = Current Supply Voltage
6: **else**
7: $V_{correct}$ = Current Supply Voltage
8: **end if**
9: Current Supply Voltage = $\frac{V_{correct}+V_{fail}}{2}$
10: **end while**
11: FFV = V_{fail}
12: MRV = FFV + RVM
13: **return** MRV

Figure 4.11: Process and Temperature Variation Aware Minimum Retention Voltage (MRV) Characterisation Algorithm at 25°C.

4.2.1 MRV Characterisation Algorithm

For each die, First Failure Voltage (FFV), is determined through voltage scaling, and then a Retention Voltage Margin (RVM) is added to FFV to get the minimum retention voltage (MRV) of each die; that is, $MRV = FFV + RVM$. The added Retention Voltage Margin (RVM), is the sum of Temperature Variation Margin (TVM) and Safety Margin (SM). Temperature Variation Margin is the worst case difference in FFV at the highest and the lowest operating temperatures for a given technology and flip-flop design when considering process variation. In this work, TVM is set to 30-mV by using the maximum FFV difference of three corner case dies (Die-2; Figure 4.7), as shown in Figure 4.8. Safety Margin is set to 2% of nominal supply voltage. In this work nominal supply voltage is 1.2-V, and therefore safety margin is set to 24-mV. Therefore the retention voltage margin (RVM) is set to 30 + 24 = 54-mV.

For each one of the test chips, MRV is determined through a characterisation algorithm (at room temperature 25°C) shown in Figure 4.11. It requires three inputs:

1. Initial Retention Voltage (IRV), as a starting point to determine FFV.
2. Voltage Scaling Resolution (VSR).
3. Retention Voltage Margin (RVM).

IRV, VSR and RVM are determined through the following criteria: Figure 4.7 shows measured results to determine the difference between first and second failure voltage

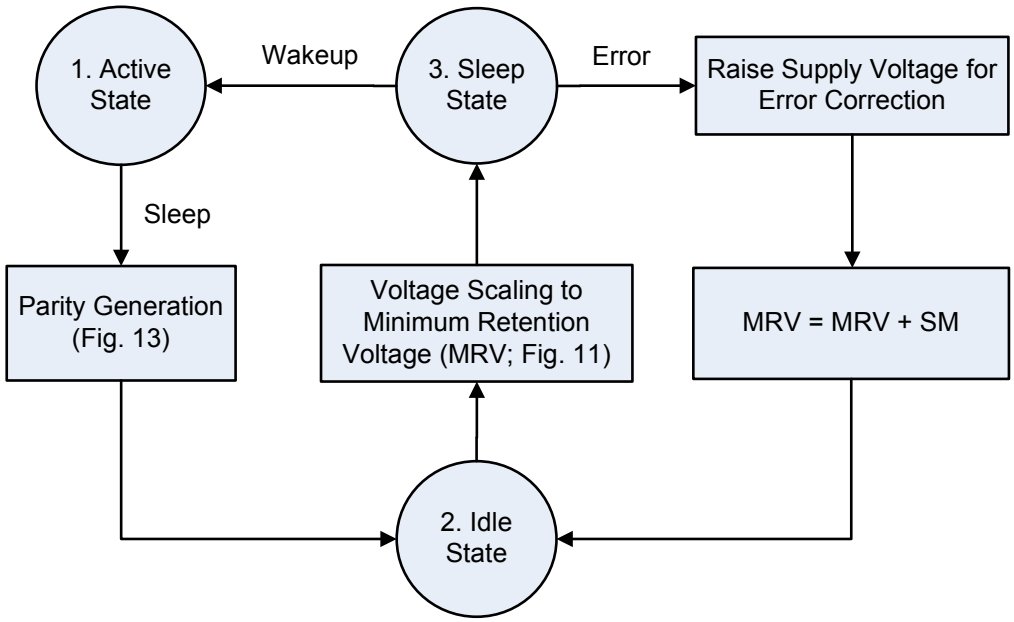


Figure 4.12: Control Flow for State Monitoring and Protection of Flip-Flops for Voltage-Scaled State Retention.

points across all test chips. These measurements are used to set the value of IRV to 400-mV. This is because none of the dies fail at this voltage. The VSR is set to 1-mV, which is the smallest step size supported by the external power supply source (Agilent U3606A). Finally, RVM is set to 54-mV to accommodate safety margin and the effect of temperature variations.

As can be seen in Figure 4.11, the algorithm starts by setting $V_{correct}$ to IRV, and V_{fail} to 0-V. $V_{correct}$ is the lower bound of supply voltage for correct state retention, and V_{fail} is the upper bound of supply voltage for failed state retention. Next, to determine First Failure Voltage (FFV), the algorithm reduces the difference $V_{correct}$ (correct state retention) and V_{fail} (failed state retention) by iterating until the difference between these two variables is smaller than VSR, which is the minimum resolution of the power supply. In line-2 of the algorithm, the current supply voltage is set to the mid-point of $V_{correct}$ and V_{fail}. In each iteration, the two variables are updated; if state corruption is detected, V_{fail} is raised to the current supply voltage, otherwise $V_{correct}$ is reduced to current supply voltage. This process is repeated by changing current supply voltage to the mid-point of updated $V_{correct}$ and V_{fail}. The loop exits with V_{fail} holding First Failure Voltage (FFV) value (line-11). Finally, the algorithm adds Retention Voltage Margin (RVM) to the observed FFV to calculate Minimum Retention Voltage (MRV) of the given test chip.

4.2.2 MRV Control Flow

The control flow is implemented using a Python script running on the host computer to communicate with the test chip through USB interface, and the test chip is powered by an external power supply (Figure 4.1-(b)). Figure 4.12 shows the control flow of the proposed technique. It consists of three states: Active State, Idle State, and Sleep State. It can be seen that as soon as 'sleep' signal is received from the host computer during "Active State", the parity is generated from the current flip-flop data and is stored in parity storage unit (Figure 4.1-(a)). Parity generation and its storage is controlled by a micro-controller (ARM C-M0) integrated in the SoC. This is why the micro-controller and parity storage unit is placed in always-on power domain (Figure 4.13). Once parity is stored, the design goes to "Idle State", after which the clock is stopped, the output of retention register block is isolated, and supply voltage is scaled down to pre-characterised Minimum Retention Voltage (MRV; Figure 4.11). The design then goes to "Sleep State". During "Sleep State", the flip-flop states are continuously monitored and compared with the stored parity bits. In the case of a mismatch, an 'Error' signal is generated in the form of hardware interrupt, which is received by the micro-controller. In response to that interrupt, the micro-controller raises the supply voltage to nominal supply voltage (1.2-V) and uses parity information (computed and saved) for single-bit error correction. In the case error correction fails due to multi-bit errors, the control software is notified through USB interface. Software state recovery such as check-pointing can be used [168]; however such well-known techniques are treated as out of the scope of this work. In the case of an error, the pre-computed Minimum Retention Voltage (MRV) is raised by Safety Margin (SM) which is set to 2% of nominal supply voltage (24-mV) to avoid subsequent errors. The updated value of MRV is stored in the host computer, which is used in subsequent "Sleep State". After increasing the MRV, the control is transferred to "Idle State", which in turn reduces the supply voltage to newly calculated MRV, and the design enters "Sleep State". Finally, upon receiving a 'wake-up' request during "Sleep State", the supply voltage is raised to nominal supply voltage, and the design enters "Active State".

4.2.3 Two-dimensional Parity for Improving Flip-flops State Integrity

Figure 4.13 shows the schematic of the retention register block that is protected using horizontal and vertical parity logic. The register block (Figure 4.1-(a)) contains 8192 flip-flops, which are divided into 8 blocks, each with ($32X32$) 1024 flip-flops. The control of the parity logic is provided by CM0 micro-controller, which is a 32-bit 3-stages pipeline RISC processor. There are two power domains (PD) in the design (Figure 4.13). Power Domain 1 (PD-1) is used for register block, which can be scaled down during state retention mode through external Power Supply. Power Domain 2 (PD-2) is used for parity storage and micro-controller, which is kept in always-on power domain (always

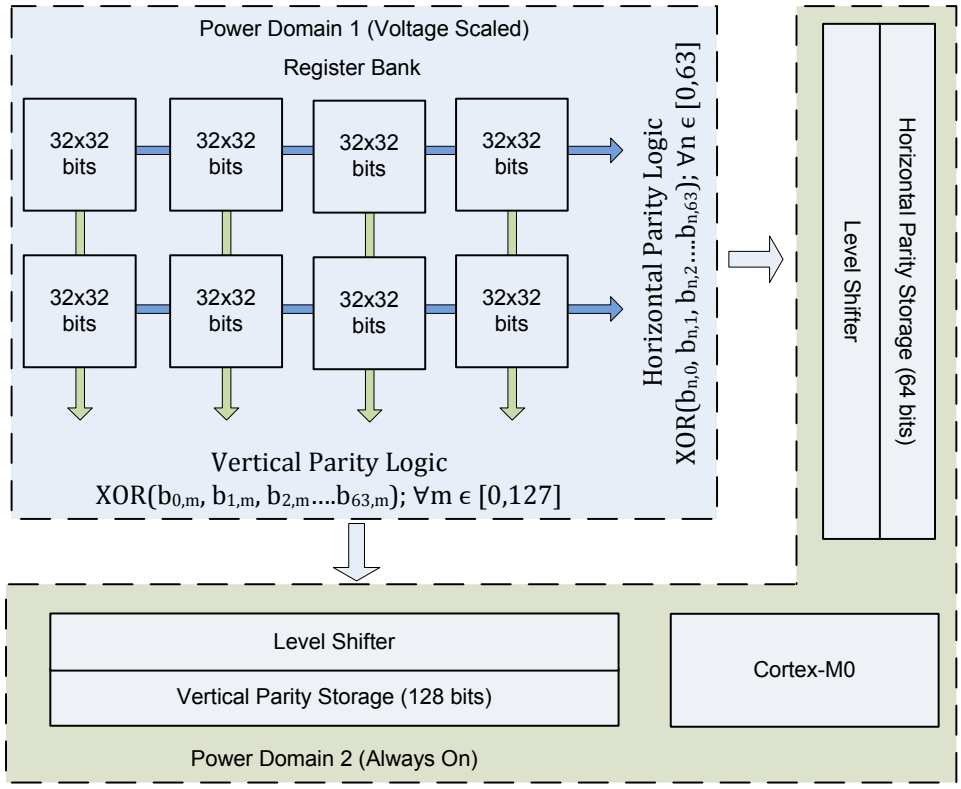


Figure 4.13: Vertical and horizontal parity protected retention register block.

operating at nominal supply voltage of 1.2-V) for continuous state monitoring of register block.

To automate the insertion of 2D parity logics, an additional step in digital circuit design flow is needed as shown in Figure 4.14. In a conventional design, firstly the RTL of a circuit is converted to gate level netlist through logic synthesis, which is followed by scan chain insertion for manufacturing test. The last stage is placement and routing. For the proposed design flow, an additional step is needed after scan chain insertion for horizontal and vertical parity insertion. This is because the scan chain converts distributed flip-flops to structured arrays. A TCL script is used to read the output of Design For Test (DFT) tool after scan chains have been inserted, and it connects all flip-flops along the scan chains for horizontal parity generation. Similarly for vertical parity generation, all flip-flops at the same depth of each scan chain are connected together. This concept is elaborated in Figure 4.14, which shows how flip-flops are connected for horizontal and vertical parity generation. Scan chains may have different numbers of flip-flops, in which case the missing flip-flop (Horizontally or Vertically) is replaced by using a direct connection. An example with two scan chains is shown in Figure 4.14, where the first scan chains has three flip-flops and the second scan chain has two flip-flops. It can be seen that the first horizontal parity is generated by using two XOR gates, while the second horizontal parity is generated by using only one XOR gate. Likewise, the last vertical parity is generated without using any XOR gate. The 2D parity logics

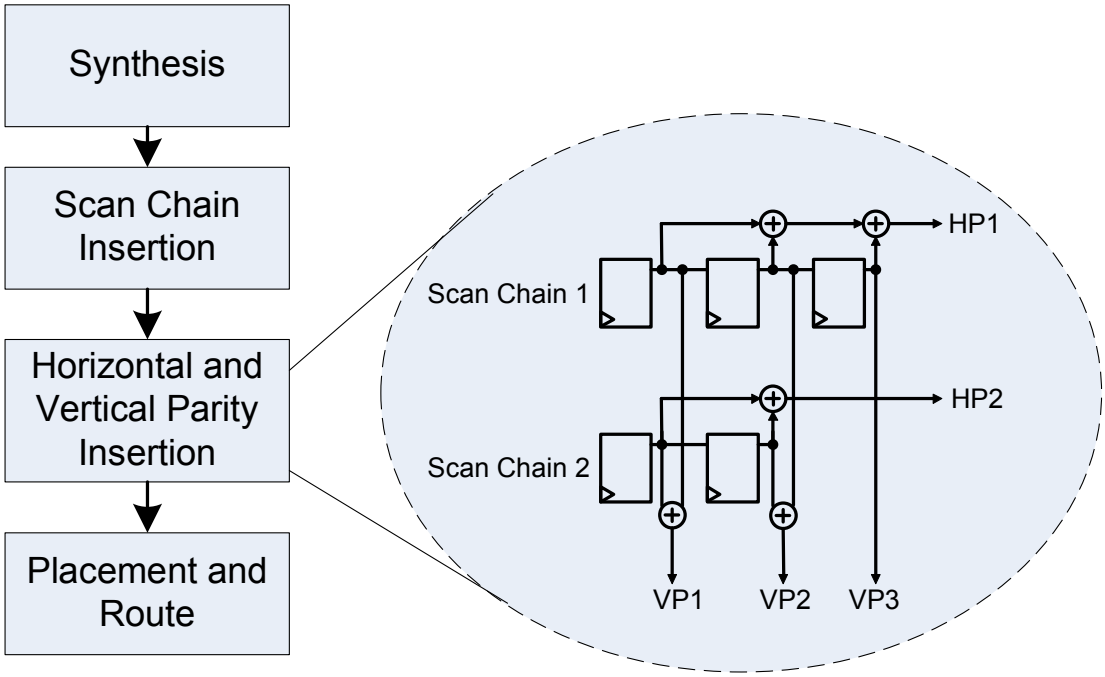


Figure 4.14: Vertical and horizontal parity insertion synthesis flow.

insertion flow is applicable for any digital circuit designs where the flip-flops organisation is not structural. For structural flip-flops such as register-banks shown in Figure 4.13, 2D parity logics were inserted into registers array in the RTL design stage.

The implementation cost of 2D parity in terms of area, power and delay needs to be analysed. The proposed technique requires two XOR gates per flip-flop. This implementation has 8192 flip-flops and the parity logic is about 51% of the flip-flop area. The parity logic used in this work incurs about 2.7 additional nets per flip-flop. From Figure 4.13, it can be observed that the number of horizontal parity storage registers is equal to the number of scan chains and the number of vertical parity storage registers is equal to the depth of the longest scan chain in the design. One level-shifter is needed for each of the parity storage registers. There is a negligible increase in delay and dynamic power in normal mode of operation; this is because the parity logic is disabled during that mode. However, the leakage power increases, which is proportional to the area overhead.

4.3 Test Chip

To study flip-flop state integrity and test the proposed PVT aware state protection technique, a test chip is designed and fabricated. Figure 4.15 shows the test chip top level architecture, which is divided into two system layers: System layer 0 (provided by our industry partner ARM Ltd) controls clock, power, communication with workstation and software program downloads; System layer 1 is for the flip-flop state integrity experiment.

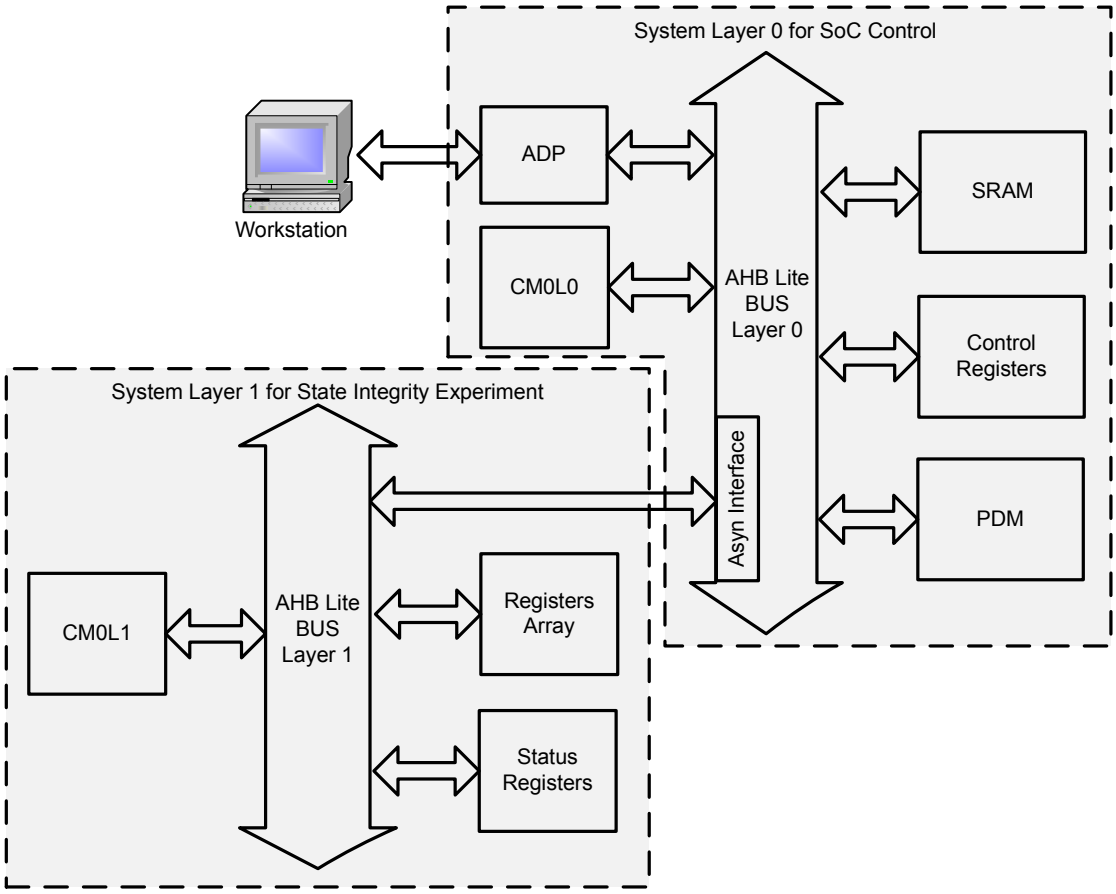
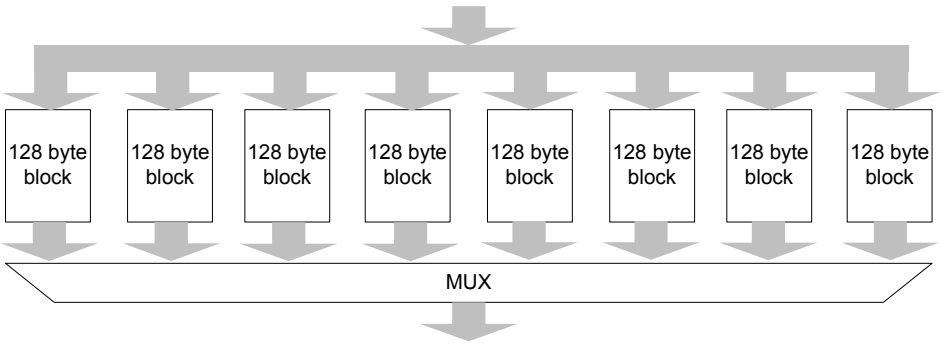


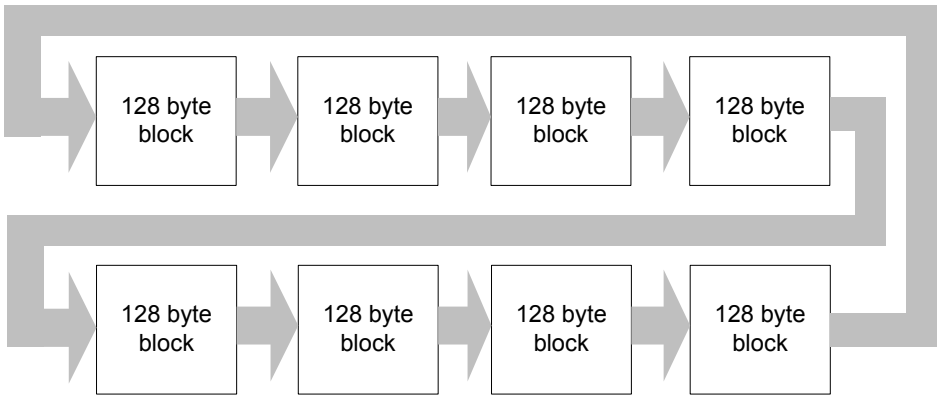
Figure 4.15: Test chip architecture.

The components are described next: components in the left-hand side of buses are bus masters and those in the right-hand side of buses are bus slaves. Bus masters only send requests and bus slaves only respond to requests, a component has read and write access to the components in its right-hand side only. ASCII Debug Protocol (ADP) provides character-based read and write access to the entire memory map; it is used to download software programs to SRAM and to send back test results to workstation. The ARM Cortex M0 in Layer 0 (CM0L0) is used for general SoC control. Control registers stores clock and power control signal. Pulse Density Modulation (PDM) of the power-switches with the feed back from the ring-oscillator is used as the on-chip voltage regulator. The Registers array is used for for flip-flop state integrity experiments. ARM Cortex M0 in Layer 1 (CM0L1) is used to control the test sequence of registers array. Status registers are used to control the function of registers array and store parity information.

The main component for flip-flop state integrity experiment is the array(s) of registers which has two modes of operations as shown in Figure 4.16. Figure 4.16.(a) shows that 8192 flip-flops are divided into 8 blocks of 32x32-bit register sub-arrays, acting as memory with single cycle synchronous write and asynchronous read. To minimise circuit area, flip-flops without asynchronous reset functionality were densely constrained in the layout; and the array had to be initialised by software. Figure 4.16.(b) shows the barrel



(a) Memory access mode



(b) Barrel rotating mode

Figure 4.16: 8192 flip-flops divided into 8 blocks register array each with 128 flip-flops, there are two mode of operations

shifting mode that allows data in register array to be rotated at clock rate, which allows a programmable switching pattern to generate controllable simultaneous switching noise. Under barrel rotation mode the 8 blocks appear as a contiguous block of shift register array.

Figure 4.17 shows the power configuration of the flip-flop state integrity experiment, two banks of registers arrays are created each with 8192 flip-flops. Both banks can be used for voltage scaled state retention or switching noise generation through barrel rotation mode (Figure 4.16.(b)). They are power-gated using header switches created between main supply rail VDD and its virtual rail (VVDD). By rapidly switching on and off the header switches, Pulse Density Modulation (PDM) is used to scale voltage on-chip. The retention voltage can be observed in the virtual rails VVDD; however there is no analogue pad available for this experiment and VVDD is not visible outside the chip. Therefore two ring oscillators are employed to measure VDD and VVDD_B0. Parity bits and control signals of both banks are stored in status registers, and level shifters are created between registers array and status registers because under retention mode the supply voltage of registers arrays is different from that of status registers. Microprocessor (CM0L1) monitors registers arrays state integrity through status register

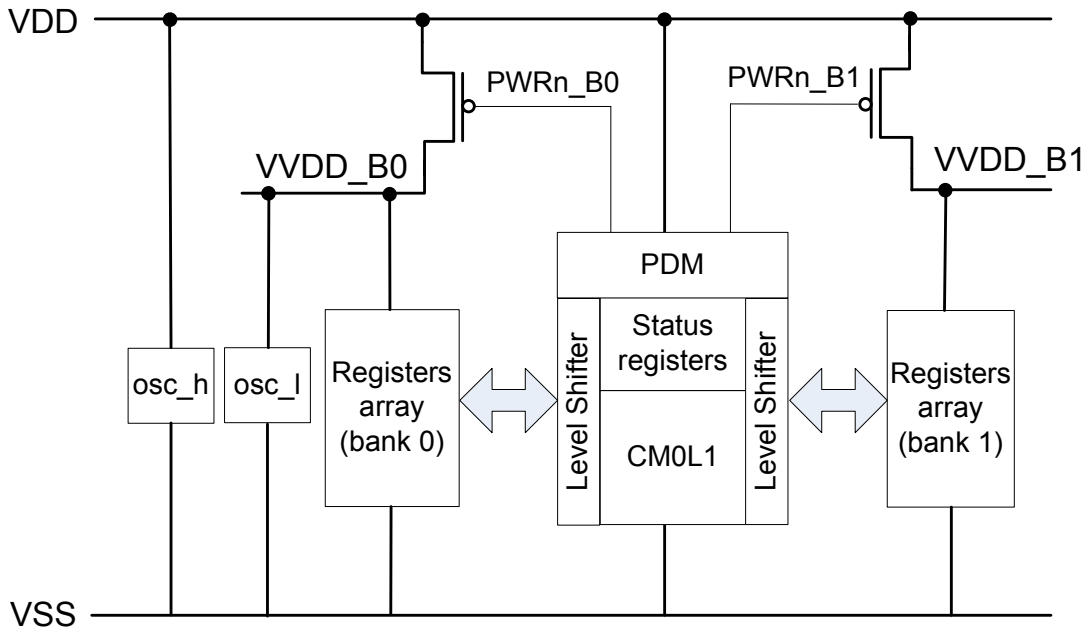


Figure 4.17: Power configuration of flip-flop state integrity experiment.

Address	Register Name	Description
0x600000000 - 0x6000003FF	SREG_B0	Registers array bank 0
0x600000400 - 0x6000007FF	SREG_B1	Registers array bank 1
0x680000000	SREG_CTRL	Registers arrays operational mode control
0x680000004	2DPAR_ERR	Parity error
0x680000008	2DPAR_INT	Parity interrupt
0x68000000C	2DPAR_PARGEN_B0	Bank 0 parity generation
0x680000010	2DPAR_PARGEN_B1	Bank 1 parity generation
0x680000014 - 0x680000020	2DPAR_VPAR_B0	Bank 0 Vertical parity storage
0x680000024 - 0x680000028	2DPAR_HPAR_B0	Bank 0 Horizontal parity storage
0x68000002C - 0x680000038	2DPAR_VPAR_B1	Bank 1 Vertical parity storage
0x68000003C - 0x680000040	2DPAR_HPAR_B1	Bank 1 Horizontal parity storage
0x680000044	OSC_H_CNT	osc_h counter
0x680000048	OSC_L_CNT	osc_l counter
0x68000004C	OSC_CTRL	Ring oscillator control
0x680000050 - 0x68000005C	2DPAR_VERR_B0	Bank 0 Vertical parity error
0x680000060 - 0x680000064	2DPAR_HERR_B0	Bank 0 Horizontal parity error
0x680000068 - 0x680000074	2DPAR_VERR_B1	Bank 1 Vertical parity error
0x680000078 - 0x68000007C	2DPAR_HERR_B1	Bank 1 Horizontal parity error
0xC000000C0	PDM_CTRL	Pulse Density Modulation (PDM) control

Figure 4.18: Memory map of components used for flip-flop state integrity experiments.

which generates hardware interrupt when parities change from the stored value. CM0L1 controls the functions of register array by setting status registers. The memory map of register array, status registers, ring oscillator control and PDM control are shown in Figure 4.18.

Figure 4.19 shows a visualisation of the power intent of flip-flop state integrity experiment. The top level power domain is *SREG_PD* whose UPF description is:

```
create_power_domain SREG_PD -include_scope
set_domain_supply_net SREG_PD
    -primary_power_net VDD -primary_ground_net VSS
```

Power supply ports are created for connections with external power supply. Supply ports are connected to corresponding supply nets which are the internal supply rails:

```
create_supply_port VDD
create_supply_net VDD -domain SREG_PD
connect_supply_net VDD -ports VDD
create_supply_port VSS
create_supply_net VSS -domain SREG_PD
connect_supply_net VSS -ports VSS
```

SREG_B0_PD and *SREG_B0_PD* are power domains for 2 banks of registers array, For demonstration purposes, only bank 0 is described:

```
create_power_domain SREG_B0_PD -elements {SREG_B0}
set_domain_supply_net SREG_B0_PD
    -primary_power_net VVDD_B0 -primary_ground_net VSS
```

The option *-elements* assigns RTL instantiation of registers array to the corresponding power domains. Two banks of registers arrays have identical internal physical structures. Supply and ground nets of *SREG_B0_PD* are created:

```
create_supply_net VDD -domain SREG_B0_PD -reuse
create_supply_net VSS -domain SREG_B0_PD -reuse
```

The option *-reuse* is to extend existing supply nets from top power domain *SREG_PD* to the hierarchical power domain *SREG_B0_PD*. Power-gating transistors are inserted between primary supply rail *VDD* and power-gated virtual supply rail *VVDD_B0*. As shown in Figure 4.17, Pulse Density Modulation (PDM) controls supply voltage of registers array by rapidly switching power-gating transistors, *PWRn_B0* is the control signal from PDM to power gating transistors. The description in UPF is:

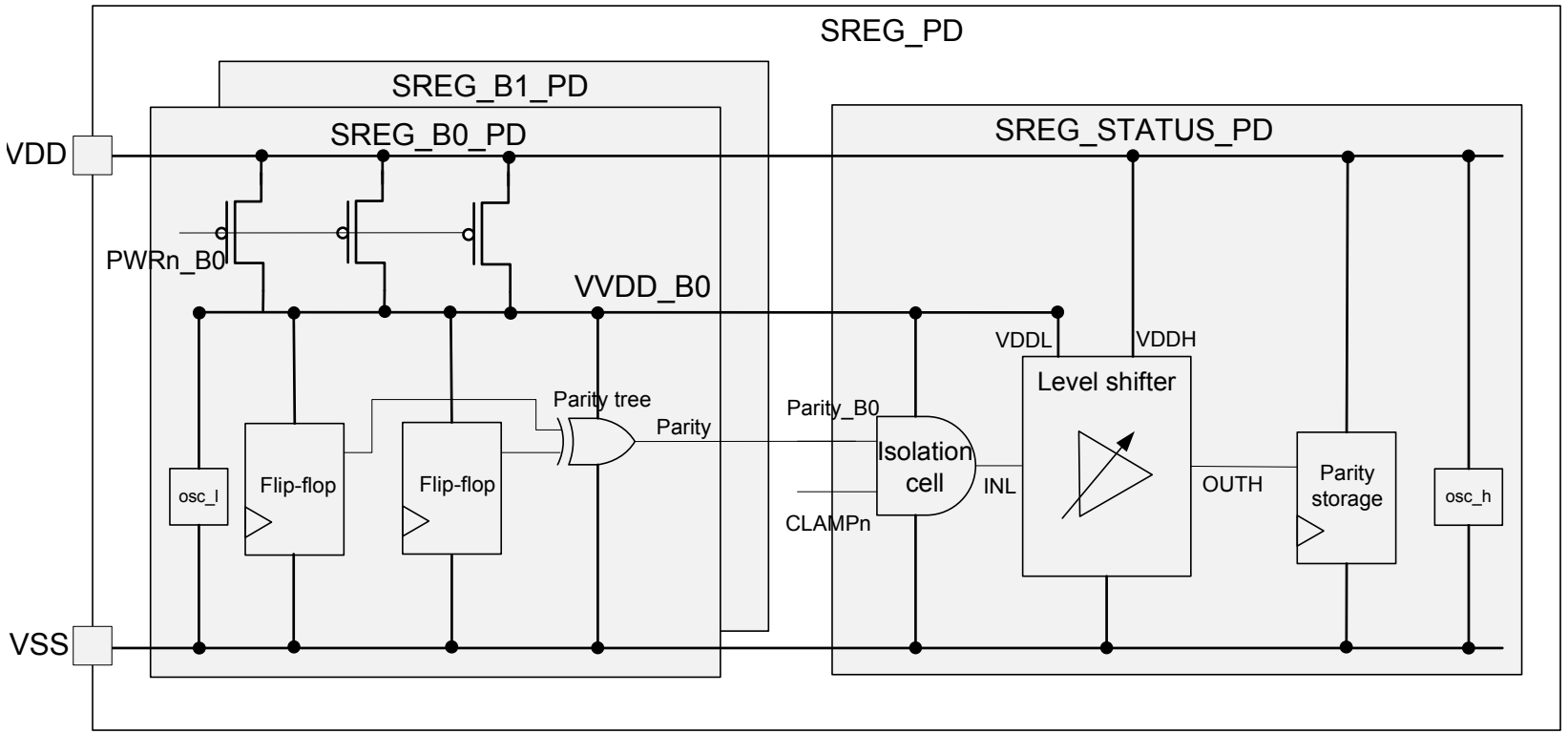


Figure 4.19: Power intent of flip-flop state integrity experiment.

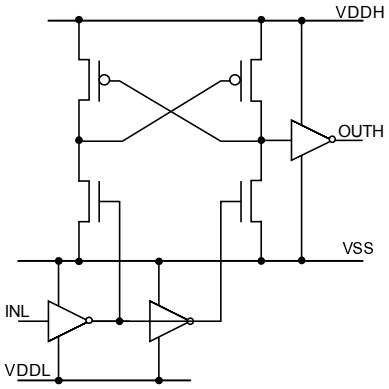


Figure 4.20: Level shifter schematic [38].

```
create_supply_net VVDD_B0 -resolve parallel
create_power_switch VVDD_B0_SW -domain SREG_B0_PD \
    -input_supply_port {VDD VDD} -output_supply_port {VVDD VVDD_B0} \
    -control_port {sleep PWRn_B0} -on_state {on_state VDD {!sleep}}
```

The parity tree is located inside register array for 2D parity generation. The output of parity chains are connected to flip-flops in status registers for storage and comparison. The comparison circuit is a simple XOR gate as comparator which is omitted in the figure. The power domain of status registers is SREG_STATUS_PD:

```
create_power_domain SREG_STATUS_PD -elements {SREG_STATUS}
set_domain_supply_net SREG_STATUS_PD
    -primary_power_net VDD -primary_ground_net VSS
```

Under retention mode, parity signals from *SREG_B0_PD* and *SREG_B1_PD* are at a different voltage level to the parity storage unit of *SREG_STATUS_PD*. In a multi-voltage design a level shifter is required especially when the signal is coming from lower voltage region to higher voltage region, this is because when the CMOS gate's input is lower than its supply voltage but higher than ground, both NMOS and PMOS transistor are half turned on creating large crossbar current [38]. Therefore level shifting is required to increase voltage level of parity signal before capturing it. The schematic of level shifter is shown in Figure 4.20 [38]. It takes a buffered and an inverted from the lower voltage signal and uses this to drive a cross-coupled transistor structure running at the higher voltage. Up-converting level shifters requires two supply rails, one from the lower voltage region of input signal and one from the higher voltage region of output signal. Therefore, beside primary supply rails VDD and VSS, virtual supply rails from SREG_B0_PD and SREG_B1_PD are extended to SREG_STATUS_PD:

```
create_supply_net VDD -domain SREG_STATUS_PD -reuse
```

```
create_supply_net VSS -domain SREG_STATUS_PD -reuse
create_supply_net VVDD_B0 -domain SREG_STATUS_PD -reuse
create_supply_net VVDD_B1 -domain SREG_STATUS_PD -reuse
```

Input isolation and level shifting are:

```
set_isolation SREG_B0_iso -domain SREG_B0_PD
    -isolation_power_net VDD -isolation_ground_net VSS \
    -clamp_value 0 -elements {SREG_B0/VPO SREG_B0/HPO}
set_isolation_control SREG_B0_iso -domain SREG_STATUS_PD \
    -isolation_signal CLAMPn_B0 -isolation_sense low -location parent
set_level_shifter SREG_B0_shift_up -domain SREG_B0_PD \
    -applies_to outputs -threshold 0.0 -rule low_to_high \
    -location fanout -elements {SREG_B0/HPO SREG_B0/VPO}
```

Signal isolation is integrated into level shifter in the cell library. Although they are defined in *SREG_B0_PD*, the physical location is in *SREG_STATUS_PD* due to option *-location fanout*. One ring oscillator is placed in *SREG_B0_PD* to measure the voltage of virtual rail *VVDD_B0* and one ring-oscillator is placed in *SREG_STATUS_PD* to measure the voltage of supply rail *VDD*.

Figure 4.21 shows the test chip synthesis and physical implementation flow. One input of the flow is Register Transfer Level (RTL) of the design written in Verilog. The other input is power intent file which defines power domains, power-gating transistors and isolation strategy of power gating techniques used in this design. The power intent is described by universal power format (UPF). The design is first validated through RTL functional simulation. After passing RTL functional tests, the design is synthesised using Synopsys design compiler and TSMC 65nm standard cell library. To fit the design into a restricted chip area, clock timing is constrained to 100-MHz to reduce excessive buffers insertion, because the focus of this study is not on high performance circuits. The synthesis stage generates gate level net list and delay description file in Standard Delay Format (SDF), which are used for post-synthesis simulation with clock speed of 100-MHz. Physical implementation of the design is done through Synopsys IC compiler. The inputs are post synthesis netlist and power description (UPF file) of the design. The physical implemenation is divided into two stages: design planning and place & route. The design planning stage includes voltage area definition, power switches placement and power grid creation. The place & route stage includes standard cell placement, clock tree synthesis and routing. Standard cell placement places the gates from the post-synthesis netlist into defined area, after which clock buffering is created to balance the clock tree to remove clock skews. Routing connects all standard cell signal lines. The placement and routing generates the final GDSII layout file that is required for

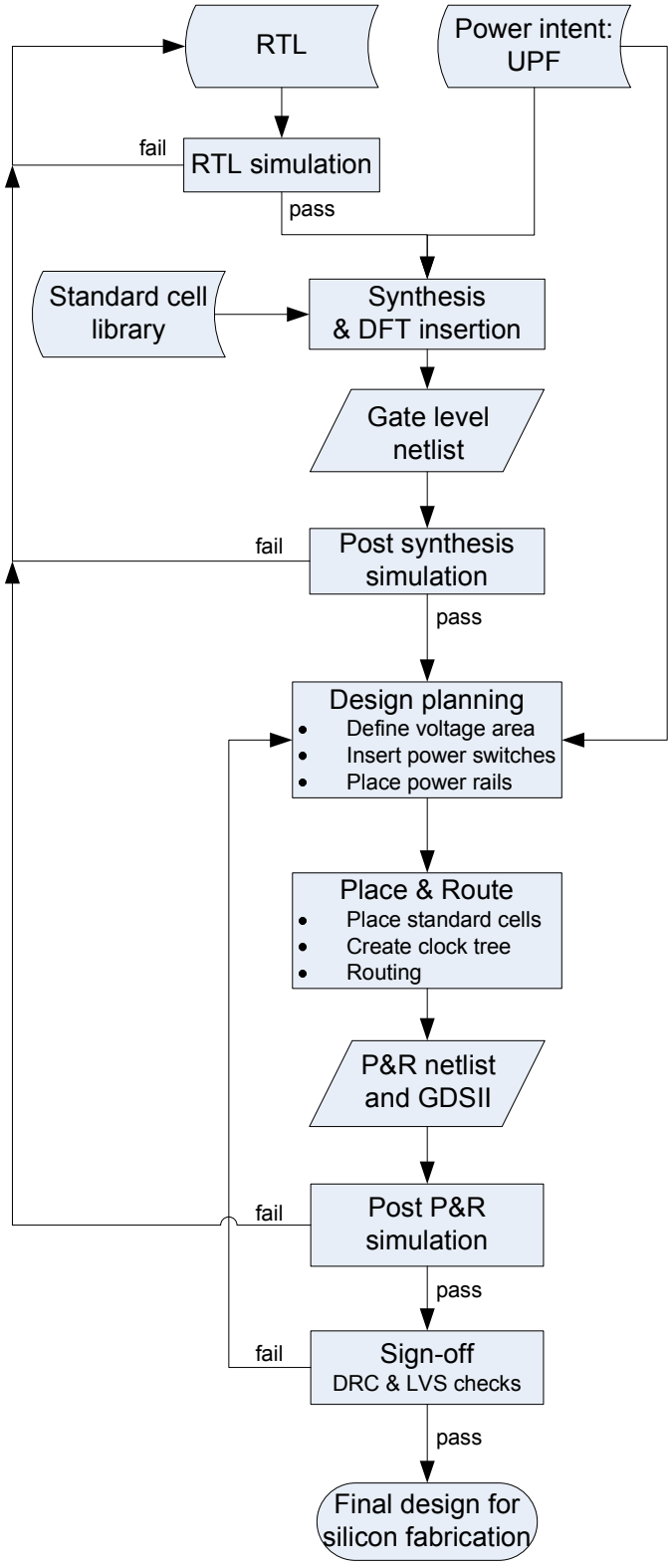


Figure 4.21: Synthesis and physical implementation flow.

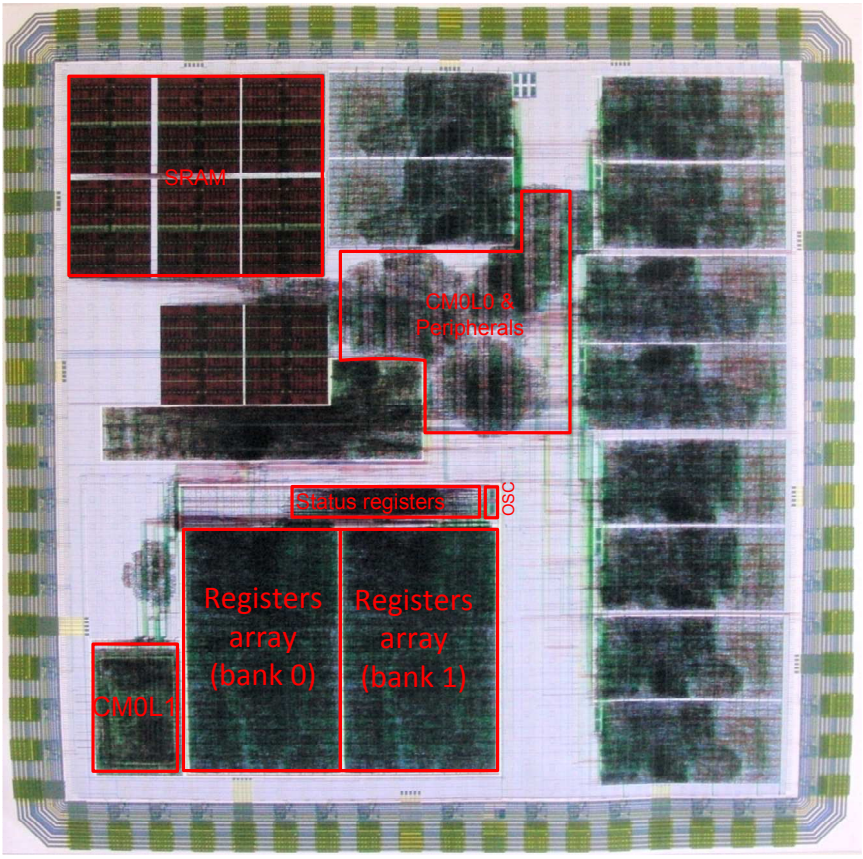


Figure 4.22: Final layout of test chip.

fabrication and delay description file for final post placement and routed simulation. Post placement and routed timing and functional simulation is done to make sure the design has been correctly constrained and implemented with extracted parasitic loads and capacitance. The final stage is silicon sign-off verification using Mentor Graphic's Calibre tool, Design Rules Checking (DRC) is done to ensure no design rule violations and Logic Versus Schematic (LVS) checking is done to ensure the function of the final schematic matches its gate level description.

The physical layout of the chip implemented using TSMC 65nm technology is shown in Figure 4.22. The flip-flop state integrity experiment only uses part of the chip area, which is highlighted on the bottom left corner of the chip. There are two blocks of register arrays each consists of 8192 flip-flops and corresponding parity check circuitry. Both blocks can be configured to study flip-flops data integrity under different supply voltage or to generate power supply noises. Status registers are implemented and used to control the register arrays and to provide parity storage. Two ring oscillators are created to monitor the voltage of supply rail and virtual rail. ARM CM0 processor core is used to set or read the status registers and react to the interrupt event when error is detected. SRAM is used to store the code and data for the software programs that control the experiment.

4.4 Experimental Results

Two experiments have been conducted to demonstrate improved state integrity of flip-flops with aggressive supply voltage scaling in "sleep state" that is possible through the proposed technique. The first experiment demonstrates improved state integrity of flip-flops in "sleep state", and the second experiment demonstrates the effect of aggressive supply voltage scaling on leakage power savings.

4.4.1 Improved State Integrity

The first experiment was conducted using three dies (Figure 4.7); the operating temperature was set to 79°C by using a temperature chamber. For this measurement, the test board was connected with a host computer through a USB interface, and Python script was used to communicate between the computer and the test board. For each die, the following five steps are repeated:

1. Voltage of the design was set to 1.2-V.
2. A single logic value (logic-1) was stored in all 8192 flip-flops, referred to as initial logic state. This is because our experiments indicate that Logic-1 state retention is about 3-times more vulnerable to bit failure than logic-0.
3. Supply voltage was reduced to respective characterised Minimum Retention Voltage (MRV) of each die for 30-minutes.
4. Supply voltage was raised back to 1.2-V.
5. Flip-flop states were compared with the *initial logic state* to determine if bit failure has been observed.

Results are shown in Table 4.1, which shows First Failure Voltage (FFV) and Minimum Retention Voltage (MRV) of each die. The fourth column shows the number of errors observed in each die, and the last column shows the response of the proposed technique. For all three representative dies, no error was detected at MRV. It is important to note that when using conventional techniques (Canary [83] and open-loop [76]), the flip-flop state is unknown. However, through this technique, it is possible to detect multi-bit errors and correct single bit error; thus it improves overall confidence on flip-flop state integrity at reduced supply voltage. When discussing measured results across 82-dies shown in Figure 4.7, it was highlighted that almost 80% of the dies exhibited only single bit failure at FFV. This is why parity logic capable of single bit error correction is used in the proposed technique to improve state integrity at reduced supply voltage. At FFV, multi-bit errors were observed in the case of just over 20% of the dies, which can be

Table 4.1: Measured results for three selected dies shown in Figure 4.7 at 79°C in "Sleep State".

Die #	FFV (mv)	MRV (mV)	# of Errors Detected	Proposed Technique Response
Die-1	315	369	0	No Error Detected
Die-2	285	339	0	No Error Detected
Die-3	250	304	0	No Error Detected

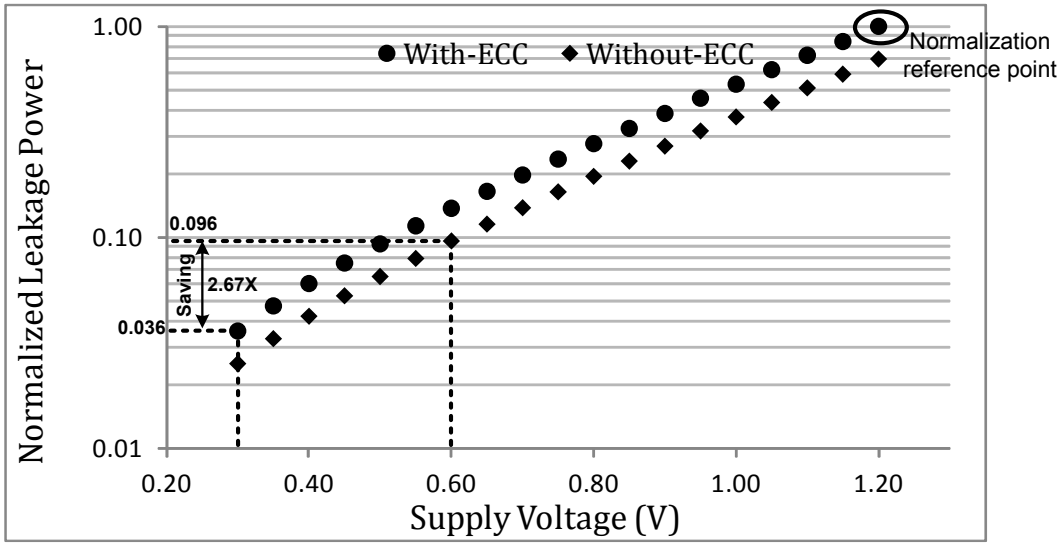


Figure 4.23: Measured Leakage Power at room temperature normalised to 1.2 V supply voltage: With ECC vs. Without ECC.

detected through the proposed technique, and can be dealt via a software check-point technique as explored in recent publications [168].

4.4.2 Aggressive Voltage Scaling

Table 4.1 shows that the difference in MRV of Die-1 and Die-3 is 65-mV, while still preserving state integrity. In comparison to using worst-case MRV across all dies, the improvement in MRV is different for each individual die. The proposed characterisation algorithm (Figure 4.11) achieves up to over 17.5% improvement in MRV in the best case (Die-1 compared with Die-3). This improvement is lower in the most common case (represented by Die-2), which is 8% lower (30-mV) than the worst-case MRV. To get an insight into potential leakage saving using this technique, a further experiment was conducted with a design without ECC, which computes Minimum Retention Voltage (MRV) across all process and temperature corners and uses that single MRV across all dies. For example, in the case of the open-loop technique, voltage is reduced to 0.6-V for all dies. On the other hand, the proposed technique employs a self-characterisation algorithm (Figure 4.11), which allows aggressive voltage scaling and each die has its own individual MRV. Figure 4.23 shows normalised leakage power with and without using

ECC. This measurement is taken at 25°C using Die-2 (Figure 4.7). As can be seen, at a given voltage, the normalised leakage power of the proposed technique is higher than that of a design without ECC. This is because of 33% area overhead of parity logic. However, the proposed technique is capable of state retention at much lower voltage leading to overall lower leakage power in "Sleep State", due to using the characterisation algorithm (Figure 4.11). For example, in comparison to a design without ECC and state retention at 0.6-V, the proposed technique can retain states at 339-mV (for Die-2) leading to 2.67-times additional leakage power savings.

4.5 Concluding Remarks

This chapter presented measured results from silicon to show that the state integrity of flip-flops is affected by process, voltage and temperature (PVT) variation. By designing and implementing special register integrity test structures and integrating these into a 65nm test chip, detailed and specific measurements have been possible to compare and contrast voltage-scaled retention techniques and approaches. Measurements from a distriction of packaged test chips, each with 8192 flip-flops, showed that even at 25°C, state integrity of a flip-flop is affected by process variation leading to spread of First Failure Voltage (FFV), from 245-mV to 315-mV, with 79% of total dies exhibiting single bit failure at FFV, while the rest show multi-bit failure. Furthermore, at elevated temperatures the variation is even more pronounced, it is found that FFV increases by up to 30-mV with increase in temperature from 25°C to 79°C. The effect of process variation with geometry scaling has also been studied using a 45-nm technology node through Monte-Carlo simulation, when compared with 65-nm technology, it is found that the overall distribution trend remains the same; however the mean FFV has shifted to a higher voltage. The effect of PVT variation on state integrity of flip-flops is addressed through development of a PVT-aware state protection technique that ensures state integrity, while minimizing state retention voltage per die. The proposed technique consists of characterisation algorithm to determine Minimum Retention Voltage (MRV) of each die, and employs horizontal and vertical parity for real-time error detection and single bit error correction. In the case of error detection, it enables dynamic adjustment of MRV per die to avoid subsequent errors. Silicon results show that at characterised MRV, the flip-flop state integrity is preserved, while achieving up to 17.6% reduction in retention voltage across 82-dies.

Chapter 5

Modelling Framework to Optimise Memory System Reliability

The previous chapters of this thesis have focused on the reliability of flip-flops, which are sequential elements of a processor core. In Chapter 3, a cost effective methodology is proposed to improve the reliability of the processor core when in sleep mode in the presence of supply noise and soft errors. Low cost protection is achieved through the reuse of scan chains for state monitoring and recovery. In Chapter 4, the state integrity of flip-flops has been studied using HSPICE simulations and measurements taken from 82 fabricated test chips. An algorithm was proposed to determine Minimum Retention Voltage (MRV) of each test chip to ensure state-integrity, while reducing leakage and state retention power. It also incorporated horizontal and vertical parity circuit for state protection under reduced supply voltage. The techniques presented in Chapter 3 and Chapter 4 have improved the reliability of the flip-flop based storage elements of processor cores. This chapter describes an architectural simulation-based framework for analysing complete processor memory system in terms of reliability, performance and energy consumption. Through the analysis, a cost effective reliable design methodology is proposed to minimise energy consumption for reliability-constrained low-power embedded processor system designs.

This chapter is organised as follows. Section 5.1 provides the motivation for this work. Section 5.2 describes the architectural simulation-based framework for estimation of memory reliability, processor performance and energy consumption. Section 5.3 shows the analysis of memory system reliability using the constructed framework. The proposed cost effective reliable design methodology and reliability aware joint optimisation is presented in Section 5.4 and Section 5.5 draws conclusions for the application of the proposed methodology.

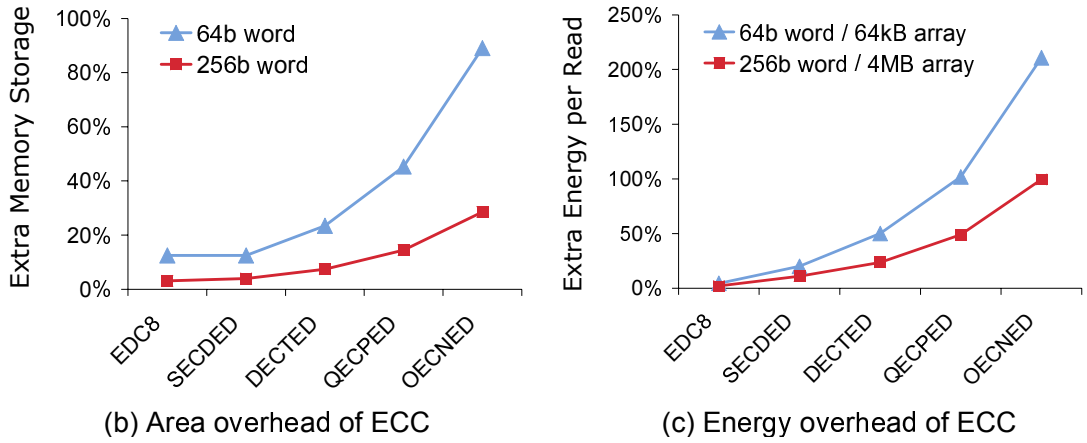


Figure 5.1: ECC overhead in terms of (a) area (b) energy [169].

5.1 Motivation

Soft errors also called Single Event Upset (SEU) are radiation-induced transient errors caused by neutrons from cosmic rays and alpha particles from material used for IC packaging. Traditionally soft errors were only a concern for space applications, but for advanced technology nodes, system level soft errors are much more frequent due to lower critical charge and higher integration density [11]. The induced charge from a particle strike will flip the state of the memory cell, which can propagate and cause erroneous outputs. Together with performance and power, reliability has become an important design parameter for many embedded processors such as micro-controllers, servers and networking devices. To protect a system against soft errors, redundancy is usually required which has a cost in chip area, power and performance [161, 169, 170]. Figure 5.1 shows Error Control Coding (ECC) area and energy overhead for various coding schemes, where the overhead increases exponentially with error correction capability. ECC protection is costly; therefore excessive protection may make the product uncompetitive in terms of cost and energy consumption. On the other hand, inadequate protection from soft errors may lead to unreliable in-field operation, which is undesirable. In a processor, the memory system is particularly susceptible to soft error because it occupies the largest fraction of the chip area [171]. Therefore, appropriate protection of a memory system is necessary, which is only possible through careful evaluation of soft error susceptibility.

There are two main types of random access memories Static RAM (SRAM) and Dynamic RAM (DRAM), which are commonly used in processor systems. SRAM is expensive but fast and it is used as cache memory to reduce access latency. DRAM requires dynamic memory cell refresh, has longer access time but is lower-cost for bulk memory and it is used as main memory to provide large memory size. A typical example of the modern processor memory hierarchy is shown in Figure 5.2, which consists of two levels of cache and a main memory. Level-1 (L1) cache normally provides both data and instruction

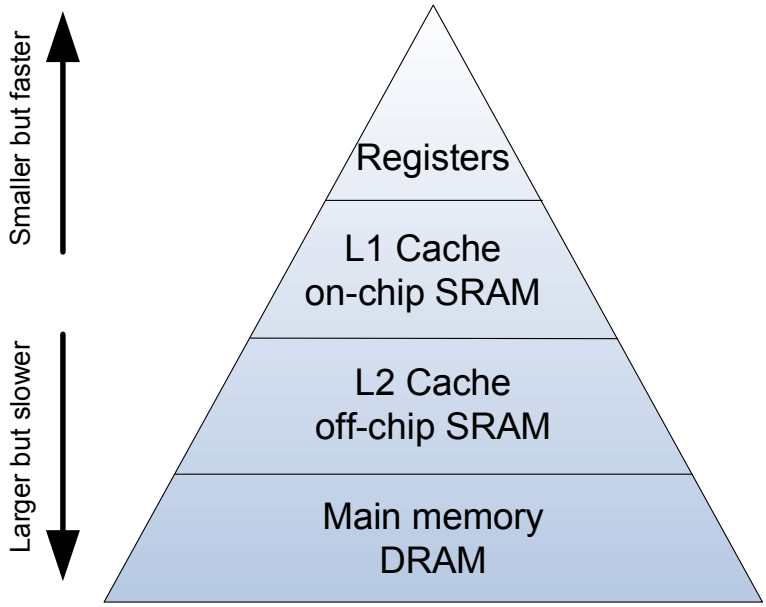


Figure 5.2: Low-medium performance microprocessor memory hierarchy

caching which connected directly to the processor core. The speed of L1-cache has the highest impact on the processor's performance and typically SRAM latency is within a few nano-seconds [172, 173]. It is usually integrated on the same die as the processor core to reduce access latency. Increasing cache size has a negative impact on access latency due to a more complex address decoder. For this reason, the maximum size of L1-cache is limited to 32kB in most modern processors such as POWER7 [174] and OMAP4460 [175]. When larger cache memory is needed, it is usually divided into two levels, a smaller L1-cache and a larger but slower L2-cache. In high performance processors level-3 cache is also used to reduce access to the main memory. The focus of this research is on low power embedded processors and therefore only two cache levels are considered together with the main memory as shown in Figure 5.2, although in some systems the L2 cache could be on-chip or off-chip.

To accurately analyse the reliability of each memory component with consideration of performance and energy consumption, an analysis framework is developed and described in the next section. This framework is used for joint-optimisation of reliability, performance and energy consumption when considering a given embedded processor using a wide variety of benchmark applications and soft error rates.

5.2 Framework for Memory System Reliability Analysis

Figure 5.3 shows the analysis framework for the application-aware joint analysis of reliability, performance and energy of an embedded processor system. The analysis framework is based on an architectural simulator, which is available as an open-source software

that models the processor system's behaviour to predict performance matrices for a given input. Such a simulator is often used in the design process to allow easy design space exploration and early software development. The architectural simulator used in this work is GEM5 [176]. It is an open source and highly configurable simulator that supports most modern processor models including ARM and x86. GEM5 is an instruction level accurate simulator but can be configured to run at near cycle accuracy at the cost of simulation speed. The inputs to this analysis framework (Figure 5.3) consist of processor system configuration files and benchmark applications. A processor system consist of hardware and software, and configuration files that define the hardware set-up such as computing resources available for processor core and memory system. Table 5.1 shows an example configuration. Benchmark applications are compiled to software binaries using MiBench benchmark suite [177] as a case study. To estimate reliability, a memory read and write access monitor is embedded into GEM5 simulator. To analyse system energy consumption an activity-based power analysis tool McPAT [178] is used, which uses the system configuration to estimate chip area and static power consumption. It uses activity statistics generated by GEM5 simulation to produce dynamic power consumption. A python script is used to post-process GEM5 statistics to be compatible with McPAT. For performance analysis, the built-in features of GEM5 have been used to generate detailed performance statistics. This simulation set up runs on an Iridis computer cluster so multiple simulations can run in parallel. A python script is used to setup the analysis framework and initiate simulations. In this case study the design space consists of 24 configurations and 24 benchmark applications, which leads to $24 \times 24 = 576$ simulations running simultaneously. Each simulation generates reliability, performance and energy consumption profiles for the corresponding configuration and application, which are used for joint optimisation. Each component of the framework is discussed in detail in the rest of this section.

5.2.1 Hardware Configuration Method

One of the inputs to the analysis framework (Figure 5.3) is the hardware system configuration. In this work, an ARM low power embedded processor is modeled using GEM5. Table 5.1 shows the processor configuration used in this work. As can be seen, it has two integer adders, one integer multiplier and divider, two floating point ALUs, one load and one store unit. It can fetch 3 instructions every clock cycle, and the issue and commit widths are 8 instructions. There are 2 levels of caches: Level-1 consist of 2-way set associative instruction and data caches with single CPU clock cycle access latency, the size of both instruction cache and data cache are 32-KB; L2-cache is 8-way set associative and has an access latency of 8ns, the size of L2-cache is 256-KB. The size of DRAM is 512-MB which has an access latency of 100ns. Hardware components are implemented as modules written in C++. A Python interpreter is embedded in GEM5 simulator so the configuration of hardware components is set through Python scripts. This approach

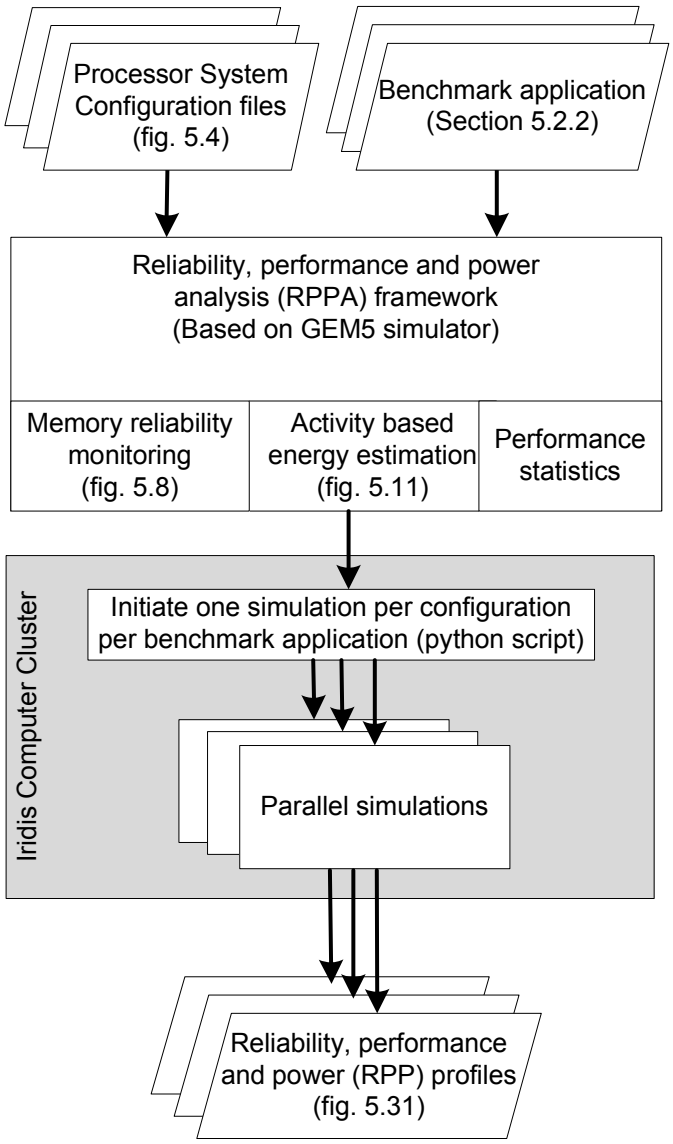


Figure 5.3: Proposed framework to analyse reliability, performance and energy consumption

combines the flexibility of Python scripting for configuration with the performance of C++ programs for execution. Figure 5.4 shows how the configuration from Table 5.1 is implemented in GEM5 using a Python script. Figure 5.4.(a) shows the configuration of functional units in the processor core, as can be seen operation latency is defined here in order to model timing information. Figure 5.4.(b) shows the configuration of the cache memory, which includes access latency, block size, associativity and size. This configuration is used for demonstration for the analysis presented in Section 5.3.

Table 5.1: System configuration used in analysis

Configuration	Value
Processor Core	
ALU	2 Integer Add/Sub
	1 integer Mul/Div
	2 float point
Fetch/Decode Width	3/3 instructions/cycle
Issue/Commit Width	8/8 instructions/cycle
Memory Hierarchy	
Instruction cache	2 ways, 1 cycle latency
Data cache	
L2 cache	8 ways, 8-ns latency
DRAM	100-ns latency

```
//O3_ARM_v7a.py

Integer Add/Sub:
class O3_ARM_v7a_Simple_Int(FUDesc):
    opList = [ OpDesc(opClass='IntAlu', opLat=1) ]
    count = 2

Integer Mul/Div:
class O3_ARM_v7a_Complex_Int(FUDesc):
    opList = [ OpDesc(opClass='IntMult', opLat=3, issueLat=1),
               OpDesc(opClass='IntDiv', opLat=12, issueLat=12),
               OpDesc(opClass='IprAccess', opLat=3, issueLat=1) ]
    count = 1

Floating point:
class O3_ARM_v7a_FP(FUDesc):
    opList = [ OpDesc(opClass='SimdAdd', opLat=4),
               OpDesc(opClass='SimdAddAcc', opLat=4),
               OpDesc(opClass='SimdAlu', opLat=4),
               OpDesc(opClass='SimdCmp', opLat=4),
               OpDesc(opClass='SimdCvt', opLat=3),
                    .....
               OpDesc(opClass='FloatMult', opLat=4) ]
    count = 2

Load/ Store:
class O3_ARM_v7a_Load(FUDesc):
    opList = [ OpDesc(opClass='MemRead',opLat=2) ]
    count = 1

class O3_ARM_v7a_Store(FUDesc):
    opList = [OpDesc(opClass='MemWrite',opLat=2) ]
    count = 1

Functional units:
class O3_ARM_v7a_FUP(FUPool):
    FUList = [O3_ARM_v7a_Simple_Int(),
              O3_ARM_v7a_Complex_Int(),
              O3_ARM_v7a_Load(),
              O3_ARM_v7a_Store(),
              O3_ARM_v7a_FP()]
```

(a) processor core

```
//O3_ARM_v7a.py

Instruction cache:
class O3_ARM_v7a_ICache(BaseCache):
    hit_latency = 1
    response_latency = 1
    block_size = 64
    mshrs = 2
    tgts_per_mshr = 8
    size = '32kB'
    assoc = 2
    is_top_level = 'true'

Data cache:
class O3_ARM_v7a_DCache(BaseCache):
    hit_latency = 1
    response_latency = 1
    block_size = 64
    mshrs = 6
    tgts_per_mshr = 8
    size = '32kB'
    assoc = 2
    write_buffers = 16
    is_top_level = 'true'

L2 cache:
class O3_ARM_v7aL2(BaseCache):
    hit_latency = 8
    response_latency = 8
    block_size = 64
    mshrs = 16
    tgts_per_mshr = 8
    size = '256kB'
    assoc = 8
    write_buffers = 8
    prefetch_on_access = 'true'
    prefetcher = StridePrefetcher(degree=8, latency = 1)
```

(b) cache memory

Figure 5.4: Example Python script for GEM5 used to configure (a) processor core (b) cache memory

5.2.2 Benchmark Applications

The second input required for the analysis framework (Figure 5.3) is one or more benchmark applications. The benchmark suite used in this work (Section 5.3) is Mibench, which assembles a set of commercially representative embedded programs [177]. The benchmark applications are written in C, and the Linaro GCC toolchain is used to cross compile the benchmark applications to ARM program binary with optimisation level 3 [179]. As shown in Table 5.2, the benchmark applications are divided into 6 groups: automotive, consumer, office, network, security and telecommunication.

1. Automotive category is intended to demonstrate the use of embedded processors in control system: "basicmath" performs simple mathematical calculations. "bitcount" tests the bit manipulation abilities of a processor by counting the number of bits in integers. "qsort" sorts a large array of strings into ascending order using quick sort algorithm. "susan" is an image recognition package for recognising corners and edges in MRI of the brain, it can also smooth an image.

2. Consumer benchmarks applications are intended for consumer devices such as scanners and digital cameras: "jpeg" is a lossy image compression and decompression program often used to view images embedded in documents. "lame" is an MP3 audio encoder with constant, average and variable bit-rate encoding. "typeset" is a typesetting tool that often used in web browsers.

3. Office category is mainly text manipulation applications: "ispell" is a fast spelling checker, it supports contextual spell checking and correction suggestions. "rsynth" is a text-to-speech synthesis program. "stringsearch" is a string searching program using a case insensitive comparison algorithm.

4. Network applications are used in network devices such as switches and routers: "dijkstra" is a routing application that constructs a graph using adjacency matrix and then calculates the shortest path between each node. "patricia" is used to maintain a tree-like data structure for routers; it collapses sparse leaf nodes to reduce the depth of the tree and therefore reduces traversal time.

5. Security category is intended for data encryption: "rijndael" is used for Advanced Encryption Standard (AES) encryption and decryption. It is a block cipher with key size upto 256-bits. "sha" is a secure hash algorithm often used in the secure exchange of cryptographic keys.

6. Telecommunication category contains the applications used in wireless communication: "crc" is an error detection algorithm widely used to ensure data integrity in transmission channels. "fft" is a digital signal processing algorithm used to find the frequencies in a given signal. "adpcm" is a compression algorithm for Pulse

Table 5.2: MiBench benchmarks [177]

Automotive	Consumer	Office	Network	Security	Telecomm
basicmath	djpeg	ispell	dijkstra	rijndael_enc	crc
bitcount	cjpeg	rsynth	patricia	rijndael_dec	fft
qsort	lame	stringsearch		sha	ifft
susan_corners	typeset				cadpcm
susan_edges					dadpcm
susan_smoothing					gsm_toast

Table 5.3: Benchmarks size

Benchmark	Instructions	Benchmark	Instructions	Benchmark	Instructions
basicmath	2,600,804,178	lame	970,864,188	rijndael_dec	437,332,579
bitcount	712,165,746	typeset	405,142,439	sha	116,047,916
qsort	323,236,108	ispell	736,368,473	crc	1,889,808,493
susan_corners	22,414,640	rsynth	1,769,029,529	fft	409,203,364
susan_edges	61,108,467	stringsearch	4,242,318	ifft	223,879,382
susan_smoothing	270,208,654	dijkstra	200,914,997	cadpcm	732,592,451
djpeg	21,833,323	patricia	591,388,089	dadpcm	526,377,589
cjpeg	95,590,130	rijndael_enc	449,518,458	gsm_toast	1,018,773,440

Code Modulation (PCM) with compression ration of 4 to 1. "gsm" is the standard of voice encoding and decoding for mobile communication.

Table 5.3 shows the number of instructions simulated for each benchmark application, which were extracted from the statistics generated by GEM5. Figure 5.5 shows the code size (the size of cross-compiled program binary) and data size (the size of input files for benchmark applications).

5.2.3 Reliability, Performance and Energy Profiling

In this chapter, the impact of soft errors on memory reliability is studied. Bit error rate (BER) is defined as the soft error rate for one storage node (bit) in a memory, represented by symbol λ_{bit}. BER is decided by the fabrication process and influenced by its operation environment. The error rates of unprotected memory can be roughly estimated using the following equation [161]:

$$\lambda_{mem} = \lambda_{bit} \times N \tag{5.1}$$

where N is the size of the memory in bits. This gives the worst case memory system failure rate estimation and over-engineering is normally used for safety-critical applications. This is based on the assumption that a system will fail if any storage node is corrupted, which is not always true. This is because some memory cells are not used

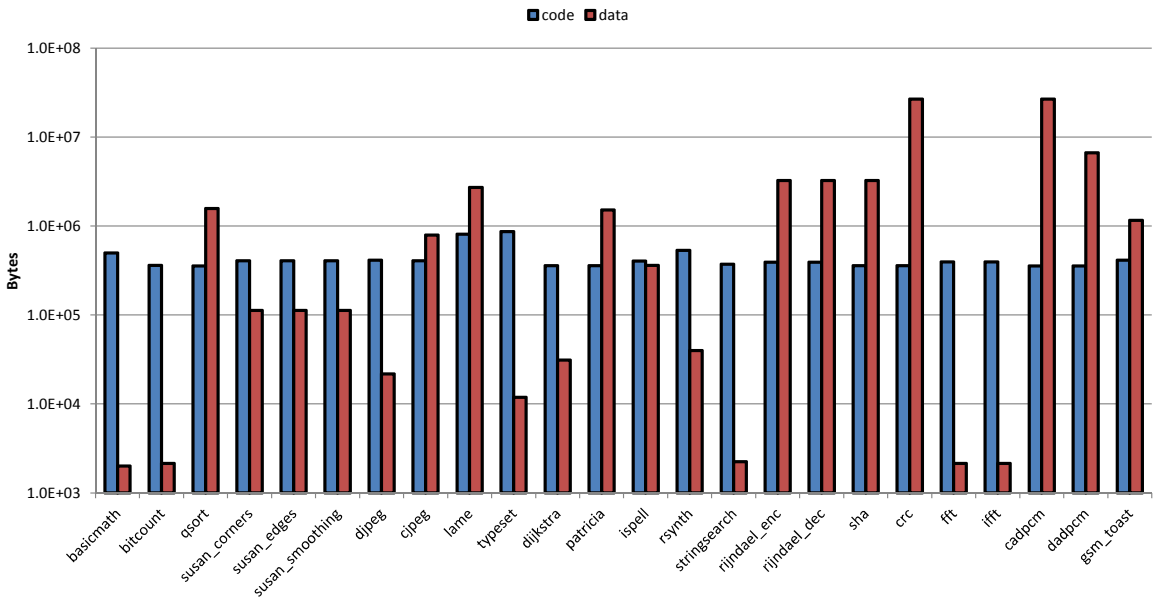


Figure 5.5: Code and data size of MiBench benchmark applications (Memory footprints)

during application runtime and even when data corruption occurs in the occupied memory cell, it may be overwritten. For example Figure 5.6 shows the data lifetime of one byte of storage cells (8 bits). At time t_0 data 'x' is written into the storage cells. At time t_1 a particle strike causes bit-flip corrupting the stored data. At time t_2 the corrupted data 'x_e' are from the storage cells and propagate to the processor core or other memory components, which may lead to erroneous output. Therefore the data lifetime between t_0 and t_2, which is vulnerable to corruption is referred to as Vulnerable Time (VT). At time t_3 data 'y' are written into the same storage cells and mask data corruption, so data lifetime between t_2 and t_3 is not vulnerable to erroneous output. Therefore, data lifetime of a memory component can be divided into vulnerable time and invulnerable time. Total vulnerable time is calculated by summing up the vulnerable time of each storage cell as shown in Figure 5.7. Equivalent vulnerable storage of an application is calculated by:

$$N' = \sum_{i} \sum_{j} VT_{ij}/Runtime \tag{5.2}$$

where 'j' represents the vulnerable time slice of one byte storage and 'i' represents storage nodes (Figure 5.7). Therefore more accurate estimation of memory reliability can be expressed as:

$$\lambda'_{mem} = \lambda_{bit} \times N' \tag{5.3}$$

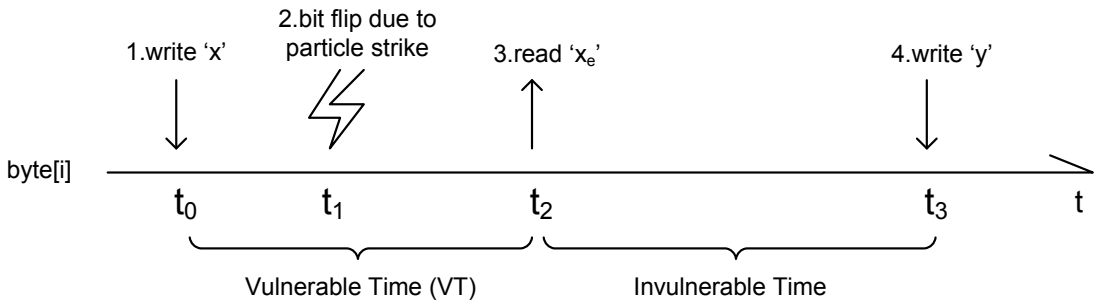


Figure 5.6: Storage cells data lifetime

where λ_{bit} is the bit error rate of the storage node. 'N'' is the vulnerable storage that can be calculated through architectural simulation of a processor system.

Vulnerable storage is used to estimate the reliability of the memory system. The amount of vulnerable storage varies across different applications. In other words, reliability is application-dependent. Architectural simulation is needed to accurately estimate the memory system reliability for a target application. GEM5 can be configured to run cycle accurate simulation, but does not generate reliability statistics directly. According to Figure 5.7 and Equation 5.2, to calculate vulnerable storage N', read and write accesses on each storage unit (the basic storage unit for memory access is one byte) needed to be monitored. Because GEM5 is an open source software, the source code of the GEM5 simulator is modified to incorporate memory access monitoring for vulnerable time collection, which will be discussed next.

Figure 5.8 shows the memory system architecture and read/write monitoring for analysing the access information of each memory component and calculating the vulnerable storage. There are read and write ports in each memory component; for the Instruction

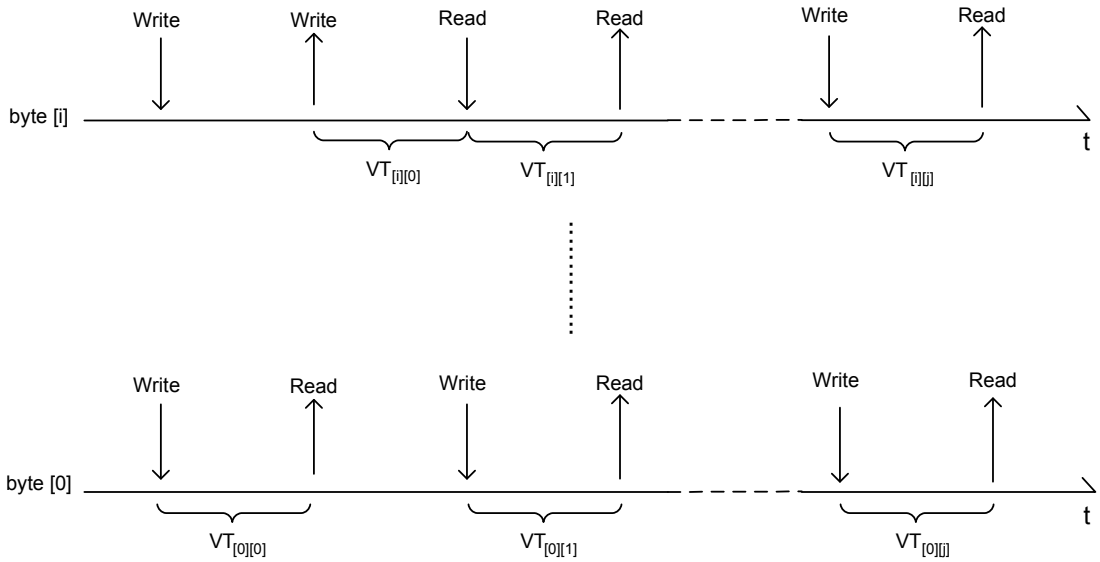


Figure 5.7: Collection of vulnerable time slices.

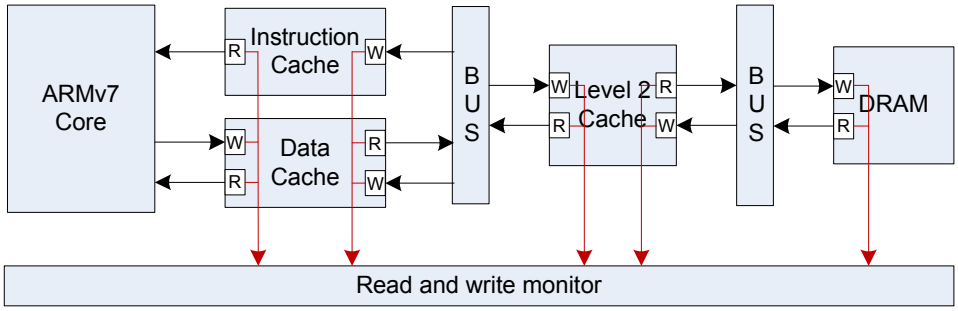


Figure 5.8: Memory system architecture and read/write access monitor

cache (I-Cache), the read port is connected to processor core to fetch instructions and write port is connected to the level-1 bus that fills the cache lines with instructions. the data cache has both read and write ports connected to processor core to enable caching of application data. Another set of read and write ports are connected to higher memory hierarchy for read and write. L2-cache provides the buffer between DRAM and L1-cache, therefore it has read and write ports connected to both level-1 and level-2 buses. The final level of memory hierarchy is DRAM which also has read and write ports connected to the level-2 bus. The read and write ports are implemented as functions in the GEM5 simulator; therefore read and write monitors are a group of functions that are embedded inside read and write ports to extract the address of memory cells accessed. Figure 5.9 shows the example code for the main memory access monitoring functions. Firstly, three variables are created: "lstAccTime" is used to store the last access time for each memory unit (1 byte of storage cells), "vulTime" is used to accumulate the vulnerable time slices (Figure 5.7), and "vulStore" is used to store the vulnerable storage calculated using Equation 5.2. Secondly, two monitoring functions are created: "writeTrace()" is used to monitor write access, and "readTrace()" is used to monitor read access. At the end of each access the monitor updates the last access time to the current time. If the access is a write operation, any error is masked and no action is needed. If the access is a read, the data are vulnerable to soft errors, and the vulnerable time (the difference between current access time and last access time) is stored into accumulator "vulTime". Lastly, the memory access monitor functions are inserted into the memory access port. Similarly, the example code for cache access monitoring is shown in Figure 5.10. Four cache accesses are monitored: write, read, cache fill and write back. Cache read and write access from the CPU is monitored in function "satisfyCpuSideRequest()", the cache fill is monitored in function "handleFill()" and the cache write back is monitored in function "writebackBlk()".

Performance statistics are generated by GEM5 directly, while power estimation is generated using McPAT to post process the dynamic activity produced by GEM5. McPAT is a power, timing and area estimation tool for the architectural simulator and the block diagram is shown in Figure 5.11. McPAT uses an XML-based interface to fetch information about hardware configuration and dynamic activity from the architectural

//abstract_mem.hh

```
class AbstractMemory : public MemObject
{
    std::map<int, Tick> lstAccTime;
    Stats::Scalar vulTime;
    Stats::Formula vulStore;
    void regStats()
    {
        vulStore = vulTime/simTicks;
    }
}
```

Variables for vulnerable time collection

Calculate vulnerable storage

//abstract_mem.cc

```
void AbstractMemory::writeTrace(Addr addr, int size)
{
    for(int i=addr; i<addr+size; i++)
        lstAccTime[i] = curTick();
}
```

Function for monitoring write access

```
void AbstractMemory::readTrace(Addr addr, int size)
{
    for(int i=addr; i<addr+size; i++) {
        vulTime += curTick() - lstAccTime[i];
        lstAccTime[i] = curTick();
    }
}
```

Function for monitoring read access

```
void AbstractMemory::access(PacketPtr pkt)
{
    if (pkt->cmd == MemCmd::SwapReq) {
        std::memcpy(&overwrite_val, pkt->getPtr<uint8_t>(), pkt->getSize());
        std::memcpy(pkt->getPtr<uint8_t>(), hostAddr, pkt->getSize());
        std::memcpy(hostAddr, &overwrite_val, pkt->getSize());
        readTrace(pkt->getAddr(), pkt->getSize());
        writeTrace(pkt->getAddr(), pkt->getSize());
    }
    else if (pkt->isRead()) {
        memcpy(pkt->getPtr<uint8_t>(), hostAddr, pkt->getSize());
        readTrace(pkt->getAddr(), pkt->getSize());
    }
    else if (pkt->isWrite()) {
        memcpy(hostAddr, pkt->getPtr<uint8_t>(), pkt->getSize());
        writeTrace(pkt->getAddr(), pkt->getSize());
    }
}
```

Insert the monitoring functions in memory access ports

Figure 5.9: Example code for main memory (DRAM) read and write monitoring

//cache.hh

Variables for vulnerable time collection

```
class Cache : public BaseCache
{
    std::map<int, Tick> lstAccTime;
    Stats::Scalar vulTime;
    Stats::Formula vulStore;
```

Calculate vulnerable storage

```
    void regStats()
    {
        vulStore = vulTime/simTicks;
    }
}
```

//cache_impl.hh

Function for monitoring write access

```
void Cache<TagStore>::writeTrace(Addr addr, int size)
{
    for(int i=addr; i<addr+size; i++)
        lstAccTime[i] = curTick();
}
```

Function for monitoring read access

```
void Cache<TagStore>::readTrace(Addr addr, int size)
{
    for(int i=addr; i<addr+size; i++) {
        vulTime += curTick() - lstAccTime[i];
        lstAccTime[i] = curTick();
    }
}
```

Insert the monitoring functions in cache access ports

```
void Cache<TagStore>::satisfyCpuSideRequest(PacketPtr pkt, BlkType *blk)
{
    if  (pkt->cmd == MemCmd::SwapReq) {
        cmpAndSwap(blk, pkt);
        readTrace(blk->blkAddr|(pkt->getAddr()&(blk->size-1)),pkt->getSize());
        writeTrace(blk->blkAddr|(pkt->getAddr()&(blk->size-1)),pkt->getSize());
    }
    else if (pkt->isRead()) {
        pkt->setDataFromBlock(blk->data, blkSize);
        readTrace(blk->blkAddr|(pkt->getAddr()&(blk->size-1)),pkt->getSize());
    }
    else if (pkt->isWrite()) {
        pkt->writeDataToBlock(blk->data, blkSize);
        blk->status |= BlkDirty;
        writeTrace(blk->blkAddr|(pkt->getAddr()&(blk->size-1)),pkt->getSize());
    }
}

typename Cache<TagStore>::BlkType*
Cache<TagStore>::handleFill(PacketPtr pkt, BlkType *blk)
{
    std::memcpy(blk->data, pkt->getPtr<uint8_t>(), blkSize);
    writeTrace(blk->blkAddr, blkSize);
    return blk;
}

PacketPtr
Cache<TagStore>::writebackBlk(BlkType *blk)
{
    std::memcpy(writeback->getPtr<uint8_t>(), blk->data, blkSize);
    writebackTrace(blk->blkAddr, blkSize);
}
```

Figure 5.10: Example code for cache (L1 and L2) read and write monitoring

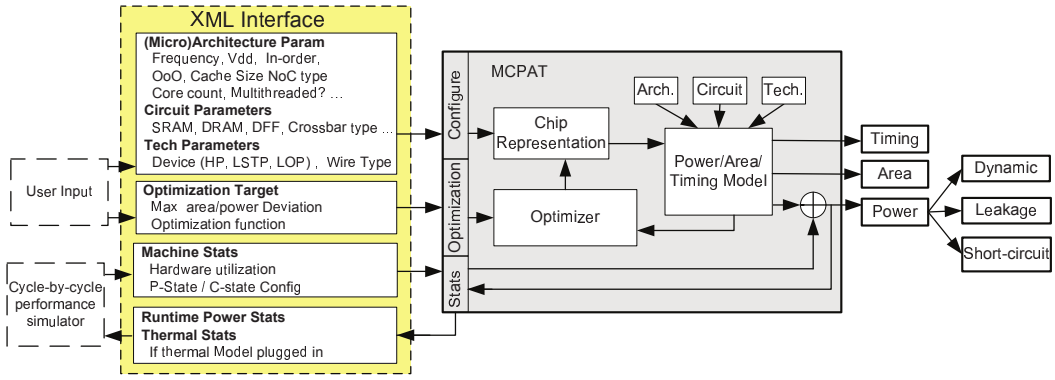


Figure 5.11: Block diagram of McPAT [178]

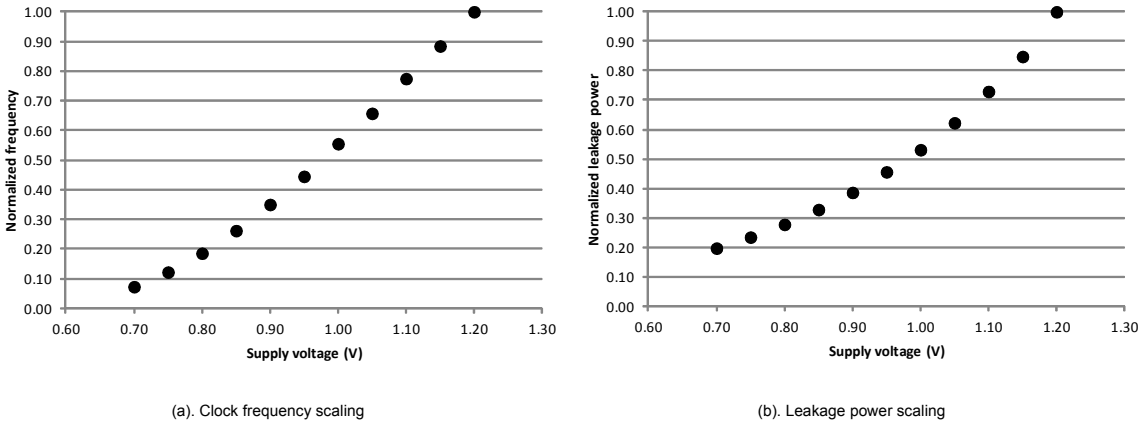
simulator. However, the output of GEM5 simulator is not in XML format, thus an additional Python script is used to convert GEM5 statistics to XML format.

5.2.4 Modelling voltage and frequency scaling

Voltage and frequency scaling (VFS) is an effective low-power design technique, which is widely used in modern processors. Various high efficiency on-chip and off-chip voltage regulators are available to support multiple power domains with different supply voltages [180, 181]. The TSMC low power 65nm technology library (used in test chip implementation in Chapter 4) uses 1.2-V as the nominal supply voltage, so in this work, it is assumed that the nominal supply voltage is 1.2-V which is reduced to 1-V, 0.85-V and 0.75-V for voltage scaling and the design is at room temperature. Further scaling of supply voltage may cause cache memory malfunction [38]; therefore 0.75-V is the minimum supply voltage chosen for this work. With additional circuitry SRAMs can be designed to operate under much lower supply voltage [182, 183]. This is beyond the scope of this work as only commercial standard cell library is considered. Table 5.4 shows corresponding processor core operating frequency, soft error rate, dynamic and leakage power at each supply voltage point, where the value are normalised to nominal supply voltage of 1.2-V. An empirical model based on the measurements from test chips (Chapter 4) is used to estimate the relationship between delay and supply voltage. From the test chip, Figure 5.12.(a) shows measurement results of normalised ring oscillator frequency under supply voltage scaling, which is used to select clock frequencies at each supply voltage point, as shown in Table 5.4. The impact of technology and voltage scaling on soft error susceptibility has been studied [111, 184], from which the corresponding soft error rate for each operating voltage is calculated. McPAT is used to parse the output of architectural simulation for power estimation [178], and the result generated by McPAT is multiplied by the power scaling ratio from Table 5.4 to incorporate voltage scaling. Dynamic power is calculated by using the following equation [185]:

Table 5.4: Normalised voltage and frequency scaling table

	1.2V	1V	0.85V	0.75V
Processor core clock [185]	1	0.5	0.25	0.125
Bit error rate [184]	1	1.7	2.56	3.34
Dynamic power [185]	1	0.347	0.126	0.049
Leakage power [74]	1	0.532	0.328	0.235



(a). Clock frequency scaling (b). Leakage power scaling

Figure 5.12: (a) Measured clock frequency under supply voltage scaling for 65nm low-power technology library, (b) Measured leakage power under supply voltage scaling for 65nm low-power technology library at room temperature.

$$P_{dyn} \propto fCV^2 \tag{5.4}$$

where f is switching activity, C is circuit capacitance and V is supply voltage. The leakage power scaling ratio is technology-dependent [74]; therefore its calculation needs an empirical model. Figure 5.12.(b) shows measurements from the same test chips (Chapter 4) describing the relationship between leakage power and supply voltage, which is used to determine the leakage power scaling ratio.

In this experiment (Section 5.3) the microprocessor (Figure 5.8) is divided into two power domains as shown in Figure 5.13: Processor core and L1-cache including instruction and data caches are located in power domain PD_CORE, L2-cache is placed in power domain PD_SOC. DRAM is normally off-chip but placed in PD_SOC for simplicity. This is because the processor core and L1-cache are usually the most power-hungry components [186], these are often located physically close to each other on the same die and shared the same power domain to meet timing at the high speed processor clock rate.

For comprehensive analysis, hundreds of simulations are needed. Architectural simulation is slow; for example, it may take days to complete a single simulation. GEM5

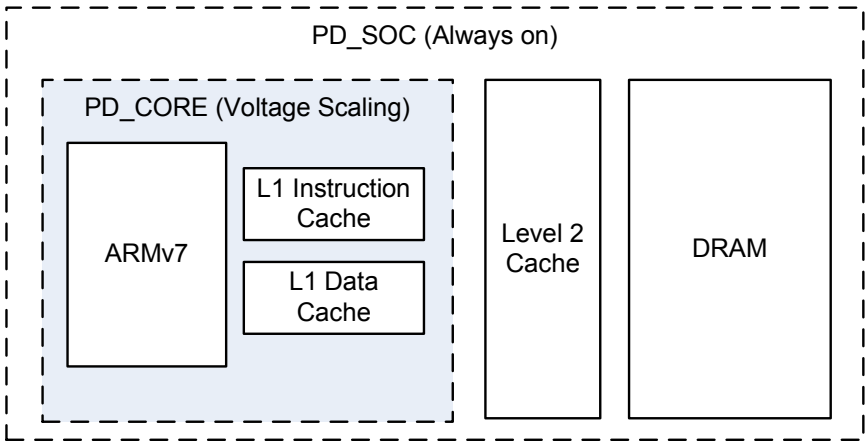


Figure 5.13: Power domains

is not multi-threaded; however, multiple simulations can be executed in parallel to reduce overall simulation time. A Python script is used to run multiple simulations on Southampton University's Iridis computer cluster simultaneously.

5.3 Reliability, Performance and Power Analysis

Using the analysis framework developed in Section 5.2, the processors' reliability, performance and power can be analysed. This analysis is used to develop a cost efficient and reliable design methodology, which is described in Section 5.4. In this section, memory system reliability is studied using error rate, which is calculated using the Bit Error Rate (BER) of memory storage cells. BER (defined in Section 5.2.3) varies between 10^{-7} to 10^{-12} depending on environment and fabrication process [152, 155, 160]. The error rate shown in this section is expressed as a multiplication of vulnerable storage (N') and BER (λ_{bit}) as described in Equation 5.3. This is because the proposed analysis framework is not limited to a specific fabrication process or environment.

Figure 5.14 shows the vulnerable storage of memory components measured across benchmark applications. It can be seen that the reliability of each memory component varies with application. For the Instruction Cache (I-Cache), the highest error rate is $1.9 \times 10^5 \lambda_{bit}$ in the case of "gsm_toast"and the lowest error rate is $1.1 \times 10^4 \lambda_{bit}$ in the case of "dadpcm". For the Data Cache (D-Cache), the error rate ranges from $2.4 \times 10^5 \lambda_{bit}$ in the case of "susan_smoothing" to $6,500 \lambda_{bit}$ in the case of "bitcount". For the L2-cache, the highest error rate is $1.8 \times 10^6 \lambda_{bit}$ when running "patricia", while the lowest error rate is $3,000 \lambda_{bit}$ when running "cadpcm". Similarly for DRAM, the error rate ranges from $2.3 \times 10^7 \lambda_{bit}$ in the case of "typeset" to $2.2 \times 10^4 \lambda_{bit}$ in the case of "stringsearch". Figure 5.15 compares the vulnerable storage of memory components. The bar shows average error rate, and the error shows the range between minimum and maximum error rate. The I-Cache has the lowest average and worst case error rate,

while main memory (DRAM) has the highest average and worst case error rate. The spread between the highest and the lowest error rate is around one order of magnitude for L1-cache and about three orders of magnitude for L2-cache and DRAM. Comparing worst case reliability, DRAM is the least reliable memory component with worst case error rate of $2.3 \times 10^7 \lambda_{bit}$, followed by L2-cache with worst case error rate of $1.8 \times 10^6 \lambda_{bit}$. This indicates that L2-cache and DRAM are more susceptible to failures in the presence of soft errors. Furthermore, the reliability of the memory system varies with software applications. Figure 5.16 shows the cumulative distribution of applications with different error rates. The x-axis shows the error rate normalised to worst case error rate, and the y-axis shows the number of applications. In the case of the I-Cache, three applications have error rates within 90% of the worst case while most applications have error rates much lower than the worst case. This spread is even greater for the D-Cache. For the L2 cache, half of the applications experience less than 10% of worst case error rate. For DRAM, 23 of applications experience less than 10% of worst case error rate. For smaller memory sizes, such as L1-cache, the spread is more even because applications tend to utilise most of the memory space. For larger memory size such as L2 cache and DRAM, applications use only part of the memory space, and the error rate is more related to the memory usage. Error rate changes significantly with application especially for larger memory such as L2 cache and DRAM. Blanket protection usually provides over-protection in terms of chip area, performance and power consumption. Cost effective protection can be achieved through fine grain protection by targeting the most vulnerable components and most vulnerable applications, this will be discussed in Section 5.4.

5.3.1 Impact of VFS on reliability, performance and energy

In a processor system, the highest level of performance is not required all the time, so there are opportunities to save power and energy through VFS when the demand for performance is reduced. Figure 5.17 shows the power distribution of the processor system under nominal supply voltage of 1.2V. The power consumption of the processor core and the L1-cache in power domain PD_CORE (Figure 5.13) dominates the system power consumption, with some variation across applications: the highest is 96% in the case of "cadpcm" and the lowest is 77% in the case of "typeset". Therefore applying voltage and frequency scaling on just the processor core and L1-cache can achieve significant power and energy saving.

Figure 5.18 shows the power reduction when supply voltage is scaled from the nominal supply voltage of 1.2V. When the supply voltage is 1V the power reduction ranges from 52% for "typeset" to 66% for "susan_smoothing". The average power reduction is 62% across applications. Further scaling of the supply voltage to 0.85V gives more power reduction of 85% on average, while the least saving is 75% from "typeset" and the

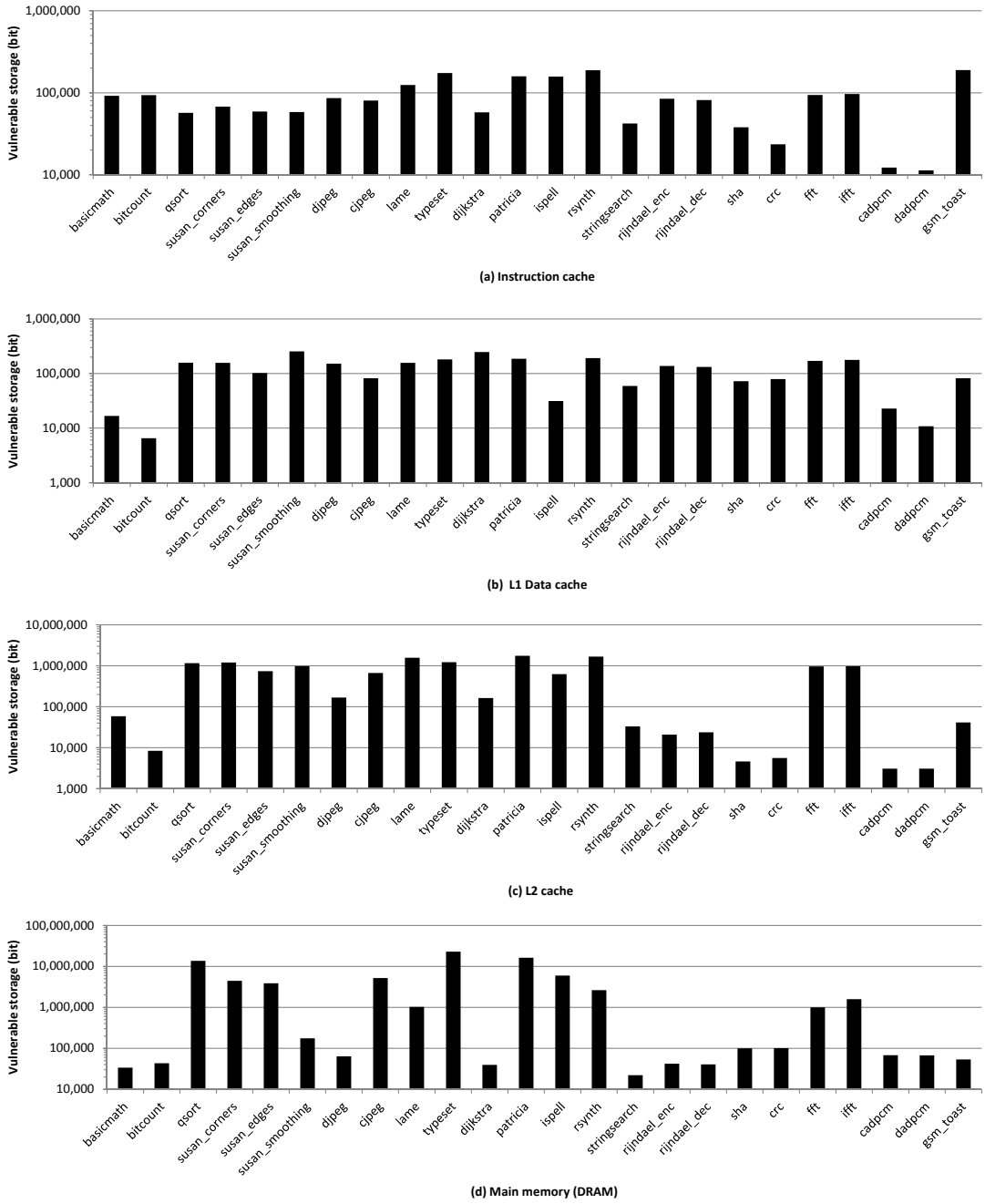


Figure 5.14: Memory components vulnerable storage

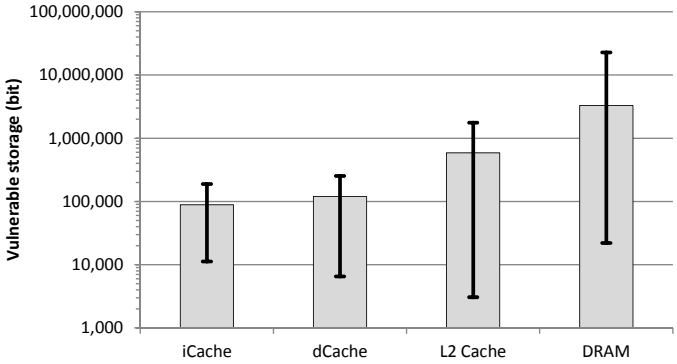


Figure 5.15: Average, minimum and maximum vulnerable storage of memory components across applications.

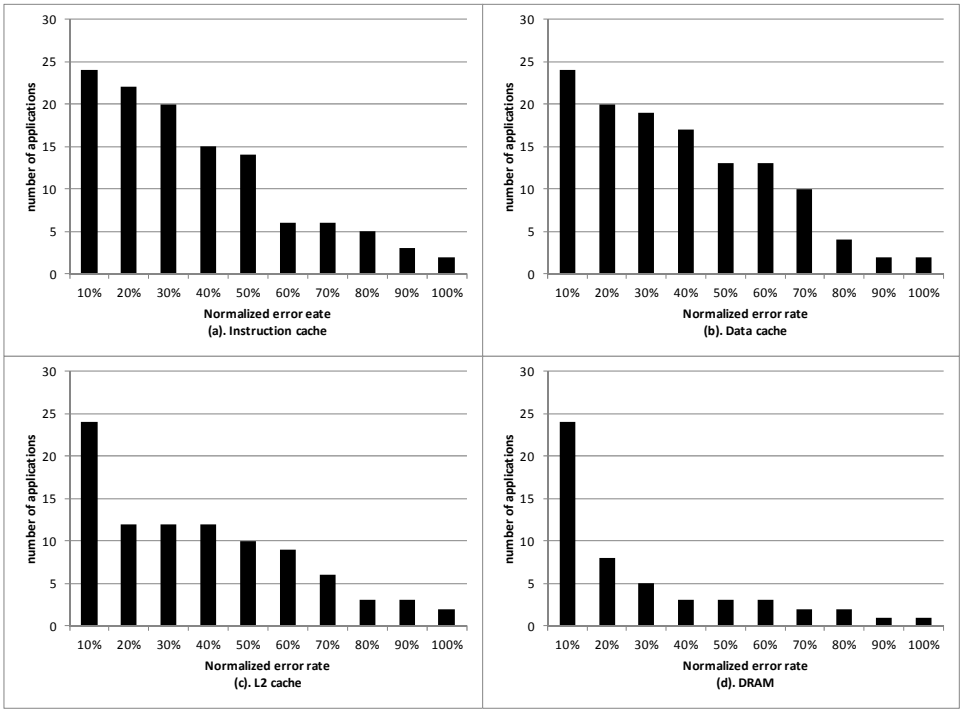


Figure 5.16: Distribution of memory component error rate for MiBench benchmark applications: (a) L1 Instruction Cache (I-Cache), (b) L1 Data Cache (D-Cache), (c) L2 cache and (d) DRAM. The error rate is normalised to the worst case error rate of each memory component.

most saving is 88% from "rijndael_enc". When the supply voltage is reduced to 0.75V, only a small amount of power saving is observed which is 90% on average. For each application, the effectiveness of VFS on the processor core and L1-cache is affected by the power distribution of power domains shown in Figure 5.17; more power saving is observed for the application where power consumption is dominated by the processor core and L1-cache (PD_CORE). Figure 5.19 shows the change of power distribution as a result of VFS. It can be seen that the percentage of PD_CORE power consumption reduces with supply voltage. At 0.75V the percentage of PD_CORE power consumption is only 56% on average, further power reduction is limited by L2-cache and DRAM power

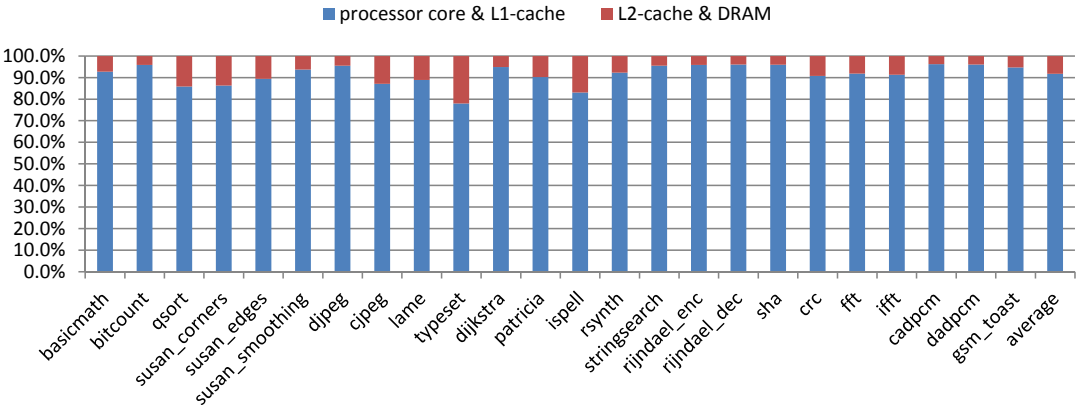


Figure 5.17: Processor system power distribution at nominal supply voltage of 1.2V

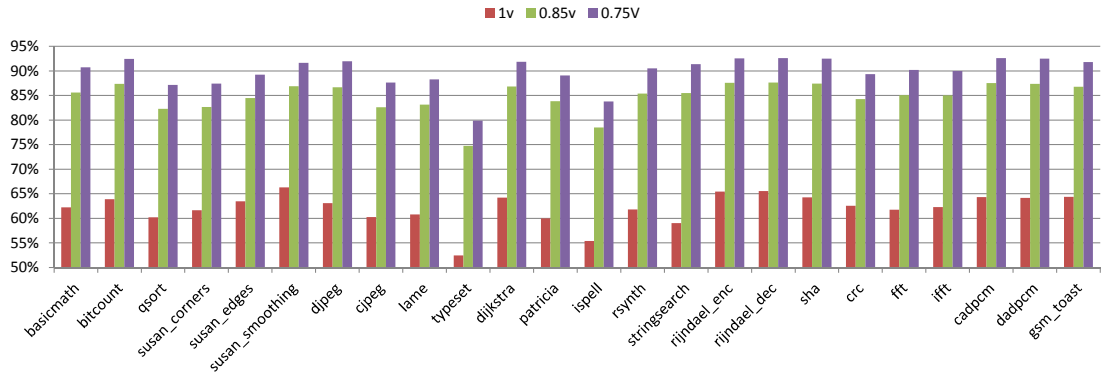


Figure 5.18: The power reduction through VFS when compared with nominal supply voltage of 1.2V.

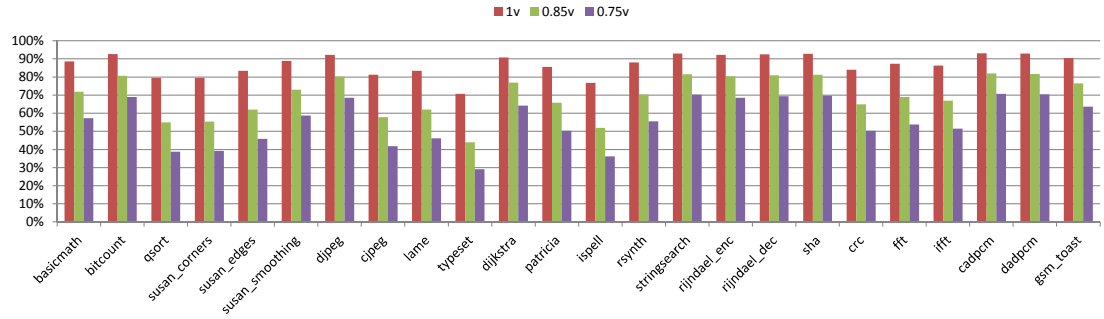


Figure 5.19: The processor core and L1-cache power consumption as a percentage of system power.

consumption.

The performance metric used in this work is Instructions Per Cycle (IPC), and higher IPC means more work done in one clock cycle. It measures the relative performance or efficiency of the processor core. The speed difference between L2-cache and L1-cache affects the performance of the processor core and creates a bottleneck for memory-intensive applications. Lower processor core clock frequency reduces the difference in L2-cache and L1-cache speed, which leads to higher IPC especially for applications

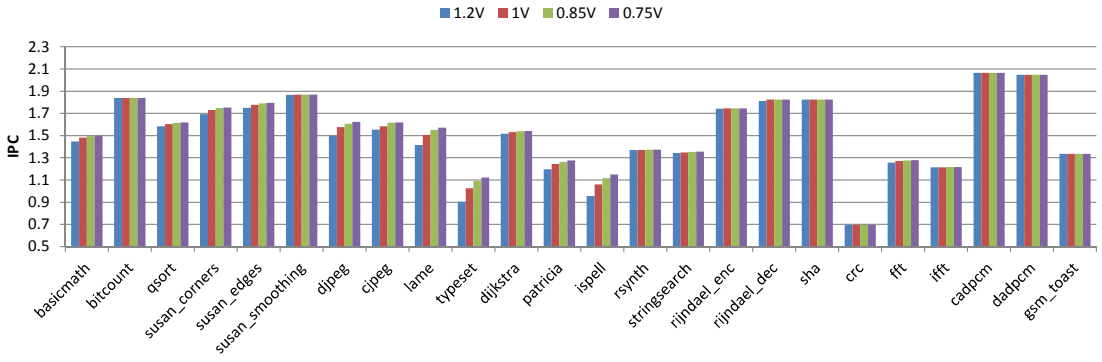


Figure 5.20: The impact of VFS on performance measured with Instruction Per Cycle (IPC).

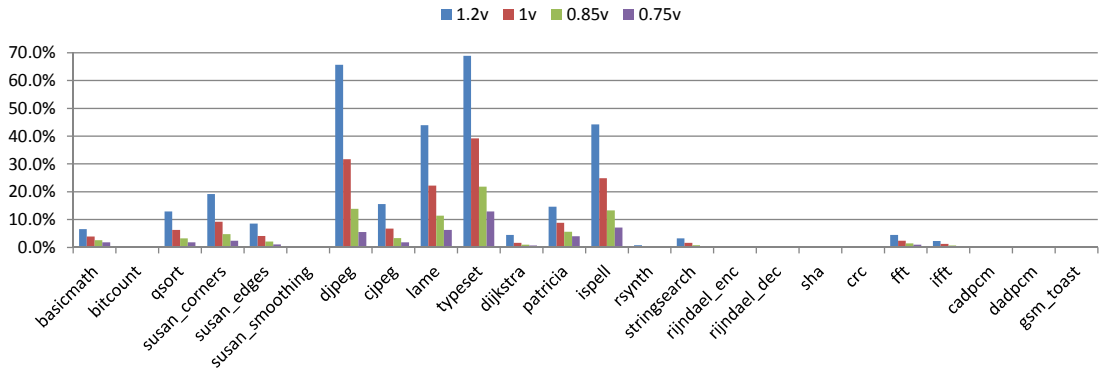


Figure 5.21: The L1-cache miss latency expressed as percentage of runtime under VFS.

demanding higher L2 cache bandwidth. Figure 5.20 shows the impact of VFS on the processor's IPC. When the supply voltage is reduced from 1.2V to 0.75V and the core clock frequency is reduced from 1GHz to 125MHz, IPC increases for applications such as "djpeg", "lame", "typeset" and "ispell". This is because the reduction in speed difference between L2-cache and L1-cache reduces the penalty for L1-cache misses as shown in Figure 5.21, where L1-cache miss latency is expressed as the percentage of application runtime. Under nominal supply voltage of 1.2V, most applications have less than 20% of L1-cache miss latency, while applications such as "djpeg", "lame", "typeset" and "ispell" spend more than 40% of their time waiting for data due to L1-cache misses. For the applications with high L1-cache miss latency, the processor core efficiency is severely affected, however the impact can be reduced by applying VFS on the processor core and L1-cache, which in turn leads to the increase in processor core efficiency.

For an energy-constrained system, minimising energy consumption is an important design goal, while ensuring that the target performance has been met. Figure 5.22 shows energy savings achieved through VFS. Energy consumption is the lowest under supply voltage of 0.85V for all applications and the average energy saving is 50%. The supply

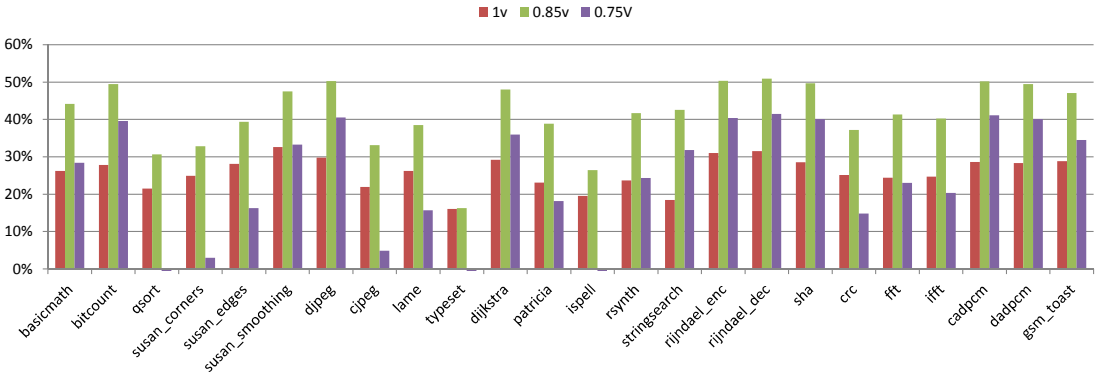


Figure 5.22: Energy reduction through VFS when compared with nominal supply voltage of 1.2V.

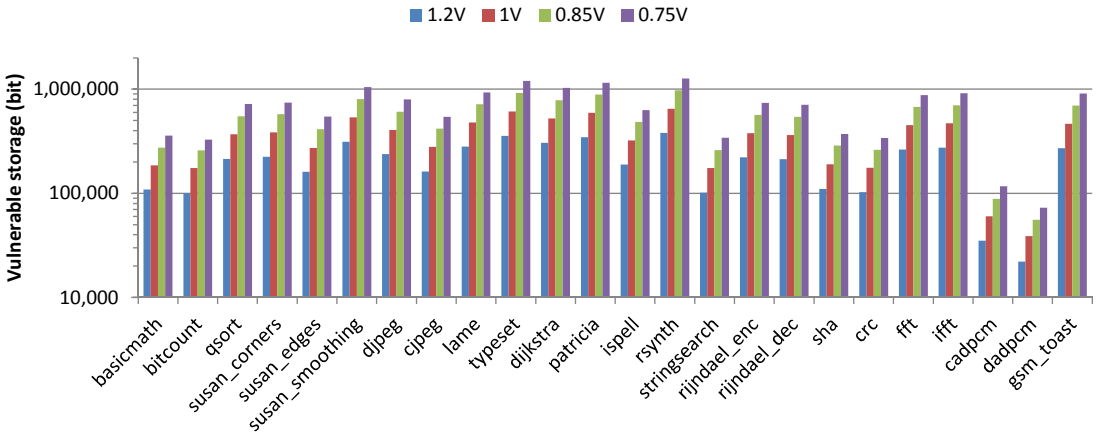


Figure 5.23: The impact of voltage and frequency scaling on L1-cache reliability.

voltage for the second lowest energy consumption varies across applications. Computational intensive applications with high IPC (Figure 5.20) have lower energy consumption at 0.75V than at 1V, such applications are "bitcout", "susan_smoothing", "djpeg", "rijndael_enc", "rijndael_dec", "sha", "cadpcm" and "dadpcm". All the remaining applications have higher energy consumption at 0.75V than at 1V.

Voltage and frequency scaling is effective in reducing power consumption and minimum energy consumption is possible through careful selection of supply voltage. However, reduced supply voltage also causes reliability problems as it lowers critical charge of the memory storage. Figure 5.23 shows, using logarithmic scaling, that the error rate in L1-cache increases across applications when the supply voltage is lowered, due to VFS on Processor core and L1-cache. L1-cache resizing is proposed to mitigate the negative impact of VFS on reliability, which is described in Section 5.4.

5.4 Cost Effective Reliable Design Methodology

There are three important observations from Section 5.3:

1. L1-cache has the lowest worst case error rate followed by L2-cache and DRAM has the highest worst case error rate (Figure 5.15). This is because data are mainly stored in a larger memory component, especially for applications with a bigger data set.

2. The error rate varies significantly with application depending on its memory access pattern. The difference in error rate across applications is 2-3 orders of magnitude in L2 cache and DRAM (Figure 5.14).

3. Voltage and frequency scaling can achieve significant energy reduction but reliability is also reduced due to reduction in critical charge of storage nodes.

These three observations are exploited to develop a cost-effective reliable design method, which is shown in Figure 5.24. The right hand side of the figure shows a conventional design implementation flow in three stages: firstly, in the design stage, the processor architecture and system configuration is based on specifications; secondly, in the implementation stage, the processor components are synthesised and integrated; and finally, software programs are loaded into the hardware system for execution. The proposed methodology is shown in the left hand side of this figure. The design of the processor system and target software applications are inputs to this framework, which incorporates reliability, performance and power analysis (RPPA) (Figure 5.3). The RPPA framework generates worst case reliability metrics of memory components. For each memory component, if the worst case reliability is lower than the requirement, hardware redundancy is inserted during the synthesis stage. The RPPA framework also generates application reliability, performance and power profile (RPPP), which can be used for dynamic protection during runtime by enabling a protection circuit only when it is needed. This section first describes how cache resizing can be used to improve reliability and secondly, it shows a case study to demonstrate joint optimisation of energy, performance and reliability.

5.4.1 Improve L1-cache Reliability through Resizing

In low power designs, the processor core and L1-cache are often subjected to voltage and frequency scaling (Section 5.2.4), which reduces L1-cache reliability (Section 5.3.1). ECC protection is a popular solution for improving memory reliability; however it is not usually a chosen solution for L1-cache because the overhead of additional delay impacts on the critical need for low latency. L1-cache resizing is proposed to reduce the error rate under reduced supply voltage without significant impact on performance and energy consumption. Figure 5.25 shows how cache resizing reduces L1-cache error rate while slightly increasing L2 cache error rate. Figure 5.25.(a) shows that the L1-cache error rate is almost halved when reducing the cache size by half. For example when cache size reduces from 32kB to 1kB, error rate reduces from $2.1 \times 10^5 \lambda_{bit}$ to $8000\lambda_{bit}$ at 1.2V

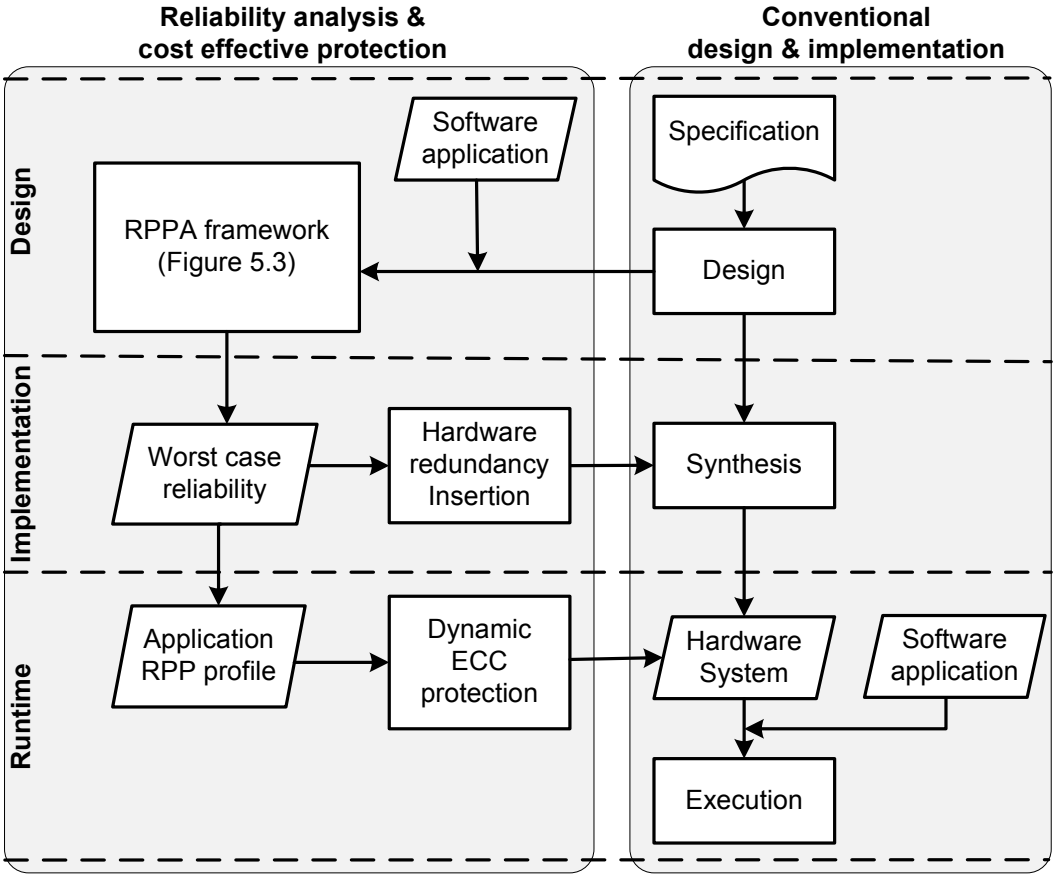
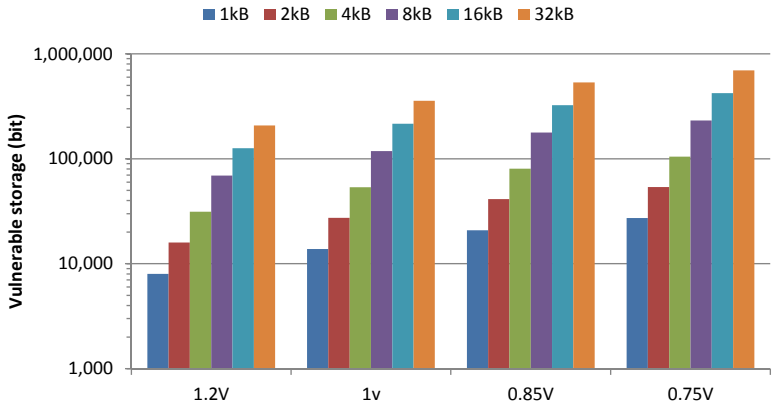


Figure 5.24: Cost effective and reliable processor system design flow.

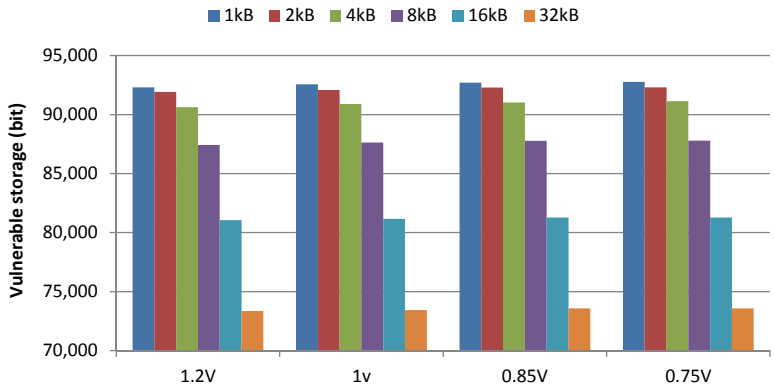
and from $7.0 \times 10^5 \lambda_{bit}$ to $2.7 \times 10^4 \lambda_{bit}$ at 0.75V. Figure 5.25.(b) shows the increment in L2 cache error rate is less than the reduction in L1-cache error rate, this is because the storage cells of L1-cache under supply voltage scaling is less reliable than that of L2 cache and vulnerable data is moved from L1-cache to L2 cache. It can also be observed that, even under nominal supply voltage settings, the increase in L2 cache error rate is slightly less than the reduction in L1-cache error rate, which is explained next.

It was shown that memory error rate is proportional to the data vulnerable time (Equation 5.2 and Equation 5.3). Figure 5.26 shows the impacts of L1-cache resizing on data vulnerable time in both L1-cache and L2 cache. In the context of cache memory, a cache line marked as *clean* means the stored data has not been modified and a cache line marked as *dirty* means the data has been modified by processor core. Figure 5.26.(a) shows when data is clean, data vulnerable time is duplicated on both L1 and L2 cache. When the cache line needs to be replaced, it simply evacuates the data from L1-cache. Therefore L1-cache vulnerable time reduces and L2 cache vulnerable time stays the same. Figure 5.26.(b) shows that in the case of dirty data, dirty cache line replacement triggers a write-back, which moves the vulnerable time from L1-cache to L2 cache. Therefore the reduction of L1-cache vulnerable time increases the vulnerable time of L2 cache.

Figure 5.27 shows that L1-cache resizing has a significant impact on performance under nominal supply voltage (1.2V), but voltage and frequency scaling on the processor core (including L1-cache) reduces the impact of L1-cache resizing on the processor core performance. This is because the impact of L1-cache miss latency reduces with the supply voltage as shown in Figure 5.21.



(a) L1-cache reliability



(b) L2 cache reliability

Figure 5.25: L1-cache resizing impact on (a) L1-cache reliability and (b) L2 cache reliability.

Figure 5.28 shows the impact of L1-cache resizing on reliability, performance and energy for different benchmark application at 0.85V supply voltage, which is the most energy efficient operating voltage (Figure 5.22) in this simulation setup. L1-cache configurations are chosen for demonstration purposes. Figure 5.28.(a) shows the the effect of L1-cache resizing on L1-cache reliability. When L1-cache size is reduced from 32kB to 8kB, there is up to 7x error rate reduction in the case of "patrica", and the average error reduction is 3x. When L1-cache size is reduced from 32kB to 1kB, the error rate reduction is up to 100x in the case of "rijndael_dec" and the average reduction in error rate is 30x. Figure 5.28.(b) shows the effect of L1-cache resizing on the performance of the processor core. When L1-cache size is reduced from 32kB to 8kB there is some reduction in performance for 13 applications; however 11 applications exhibit negligible reduction in performance. When L1-cache size is reduced from 32kB to 1kB the impact

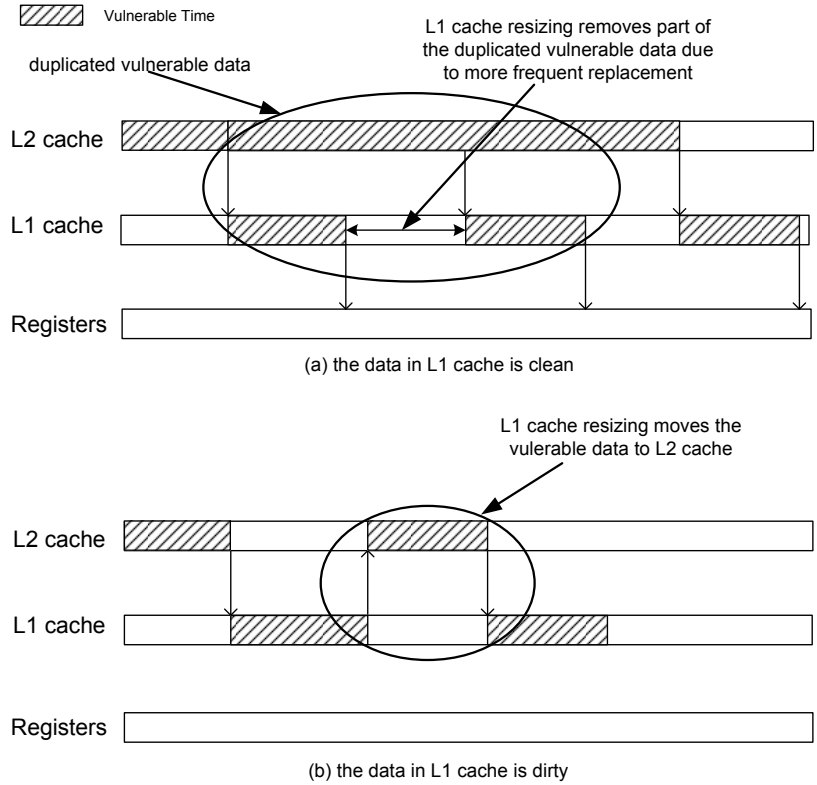


Figure 5.26: The impact of L1-cache resizing on vulnerable time.

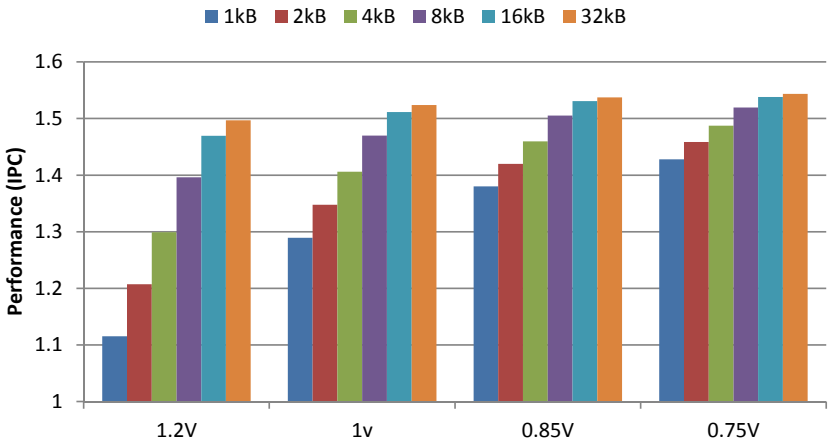


Figure 5.27: The impact of L1-cache resizing on performance.

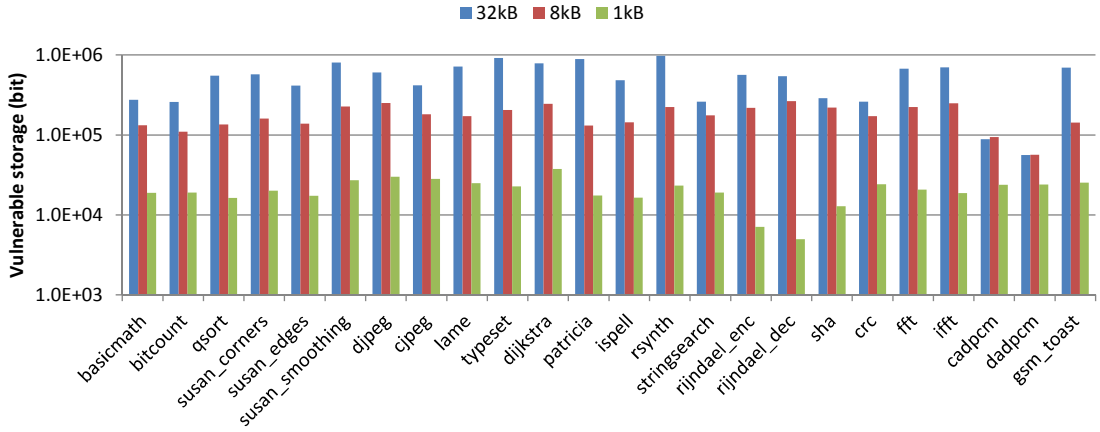
on performance is higher, but for 5 applications this impact is still negligible. As can be seen, 1kB of L1-cache is not sufficient and 8kB of L1-cache is a better choice for most applications to maintain processor performance under reduced processor core clock speed. The impact of L1-cache resizing on performance also depends on the supply voltage. At 1.2V nominal supply voltage, the clock frequency of the processor core is 1GHz, the L1-cache latency is 1ns and L2-cache latency is 8ns, and the processor core needs to wait on average 8 clock cycles for each L1-cache miss. When the supply voltage of a processor core (including L1-cache) is reduced to 0.85V, clock frequency

of the processor core must be 250-MHz, the L1-cache latency increases to 4ns and L2-cache latency stays at 8ns. Each L1-cache miss requires only two clock cycles. The impact of L1-cache miss on performance is reduced so smaller cache size does not cause significant performance loss. Figure 5.28.(c) shows the effect of L1-cache resizing on energy consumption. When L1-cache size is reduced from 32kB to 8kB the impact on energy consumption is small (up to 13%); this is because L1-cache resizing has little impact on both performance and power consumption at 0.85V supply voltage. However when the L1-cache size is reduced to 1kB, the energy consumption increases significantly (up to 60%). When voltage scaling is used on the processor core, L1-cache resizing has little impact on performance and energy consumption for some applications. This is because the penalty of L1-cache miss is relatively lower when the difference in speed is reduced. At the same time reducing L1-cache size can increase its reliability and mitigate the negative impact on reliability due to voltage scaling.

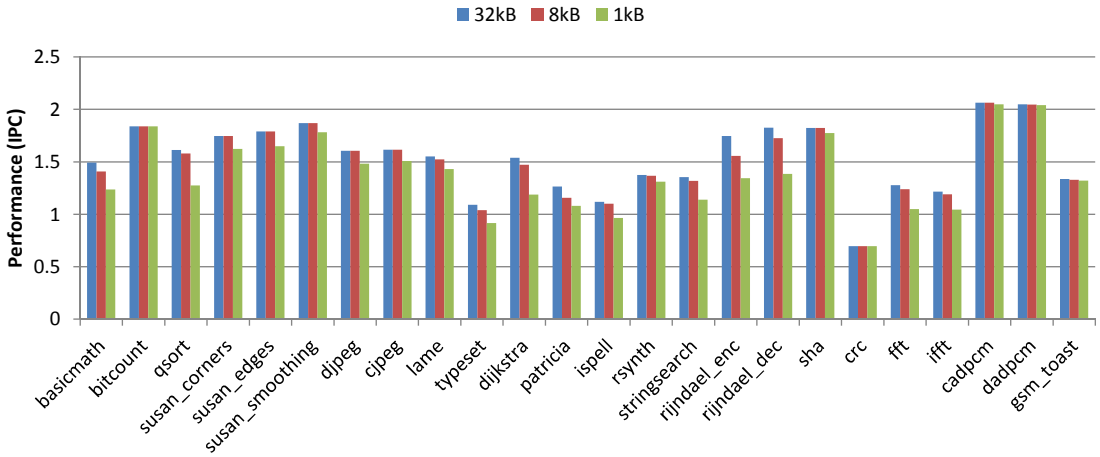
Figure 5.29 shows L1-cache resizing can be used together with voltage and frequency scaling to reduce the energy consumption while maintaining reliability. Figure 5.29.(a) shows energy consumption of a system when subjected to VFS and L1-cache resizing. As observed earlier, VFS is effective in reducing energy consumption for all applications. When L1-cache resizing is used, energy consumption increases slightly; however when combined with VFS, energy reduction is on average at about 50%. Figure 5.29.(b) shows reliability of L1-cache under VFS and L1-cache resizing. VFS has a negative impact on reliability of L1-cache, but with L1-cache resizing this reliability is restored for most applications.

5.4.2 Energy Minimization through Dynamic ECC Protection and Cache Resizing

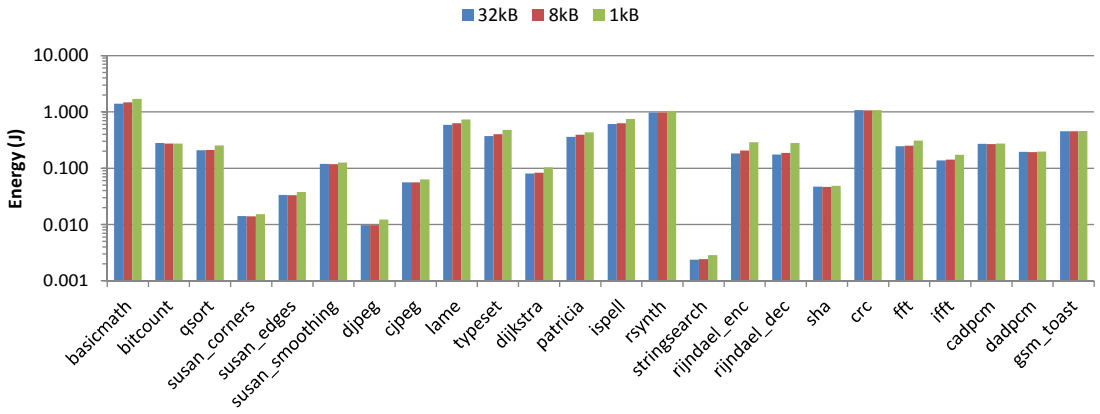
This section demonstrates the benefit of using the proposed methodology (dynamic ECC protection and cache resizing) for reliability and performance-constrained designs in terms of energy savings. It was shown in Section 5.3 that DRAM is the least reliable memory component followed by L2 cache. Therefore we assume that the ECC code used for DRAM is stronger than L2 cache. For demonstration purposes, DECTED code with 50% energy overhead is used for DRAM and SECDEC code with 25% energy overhead is used for L2 cache, the energy overhead is derived from Figure 5.1. L1-cache is protected by resizing as discussed in Section 5.4.1. Figure 5.30 shows the reliability and performance constrained energy optimisation algorithm by using dynamic protection, cache resizing, voltage and frequency scaling. As can be seen the inputs are: Reliability, Performance and Power Profile (RPPP), Dynamic Timing Constrains (DTC) and Dynamic Reliability Constrains (DRC). It is assumed that the dynamic timing and reliability constraints does not exceed the design time constraints. The RPPP is generated by the analysis framework (Figure 5.3) for each application under different



(a) Reliability

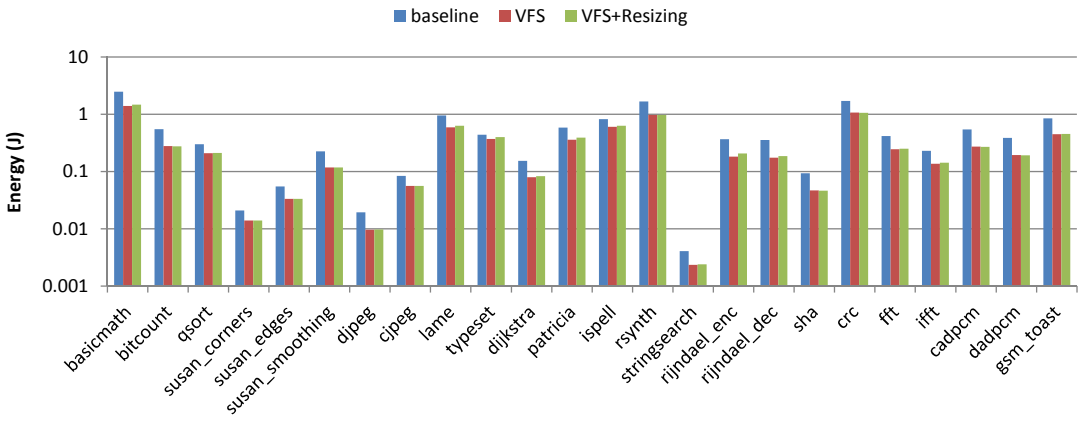


(b) Performance

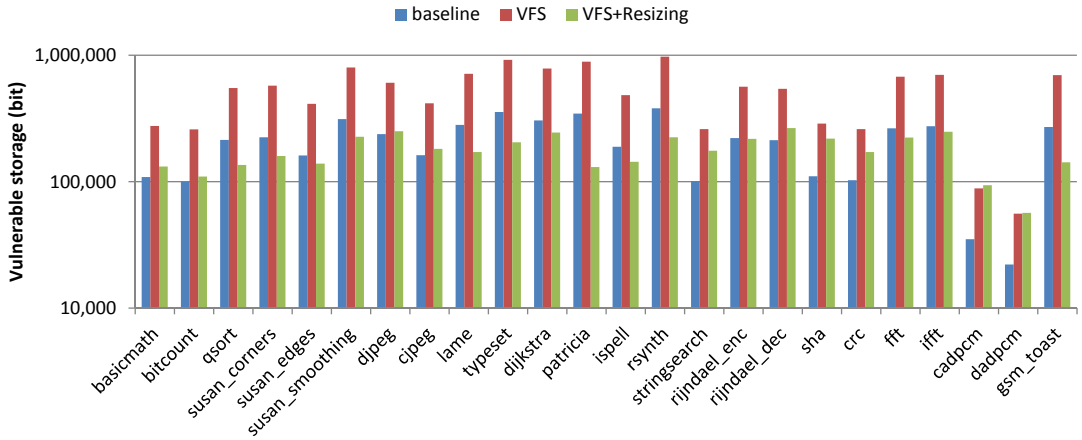


(c) Energy

Figure 5.28: The impact of L1-cache resizing on (a) reliability, (b) performance, (c) energy under 0.85V supply voltage across applications.



(a) Energy



(b) Reliability

Figure 5.29: Combined effects of VFS and L1-cache resizing on (a) energy (b) reliability.

supply voltage and L1-cache size. Some example RPPP used in this case study are shown in Figure 5.31. The algorithm begins by setting the supply voltage to 1.2V, and checks whether the processor meets the timing constraint. The reliability of L1-cache is checked next and L1-cache size is reduced until L1-cache passes the reliability constraint. The example implementation of the optimisation algorithm is described in Appendix E.

Similarly, reliability of L2-cache and DRAM is checked against reliability constraints. If it fails, dynamic ECC protection is enabled by setting $L2EccEn = 1$ and $dramEccEn = 1$ for L2 cache and DRAM respectively. The ECC energy cost is added to the overall energy. If the energy cost is minimum, then the current configuration is stored. Note that at this point, processor timing constraint is checked again because cache resizing may effect processor performance. This is followed by reducing the supply voltage and this process is repeated (line 2 to line 20) until the most energy efficient operating point is found. Figure 5.32 shows the example of joint optimisation for application "lame". Supply voltage and L1-cache size influences runtime (x-axis), L1-cache error rate (y-axis)

Input: Reliability, Performance and Power profile, Dynamic Timing Constrain (DTC), Dynamic Reliability Constraint (DRC)
Output: Minimum Energy (minEnergy), Supply Voltage (vdd), L1-cache size (L1Size), L2 ECC enable (L2EccEn), DRAM ECC enable (dramEccEn)

```
1: supply voltage = nominal supply voltage
2: while processor pass DTC check do
3:   while L1-cache fails DRC check do
4:     reduce L1-cache size
       // Ensures L1-cache satisfies reliability constraint
5:   end while
6:   if L2 cache fails DRC check then
7:     L2EccEn = 1
8:     Energy = Energy + L2 cache ECC energy
9:   end if
10:  if DRAM fails DRC check then
11:    dramEccEn = 1
12:    Energy = Energy + DRAM ECC energy
13:  end if
14:  if Energy < minEnergy AND processor pass DTC check then
15:    minEnergy = Energy
16:    L1Size = current L1-cache size
17:    vdd = current supply voltage
       // Store the configuration with minimum energy consumption while meeting
       timing and reliability constraints
18:  end if
19:  reduce supply voltage
20: end while
21: return cSize, vdd, L2EccEn, dramEccEn
```

Figure 5.30: Reliability and performance constrained energy optimisation algorithm.

and energy consumption (color coded). The rectangle defines the region of timing and reliability constrains, the configuration with the lowest energy consumption is selected inside the constraints rectangle (darker color represents lower energy consumption).

The proposed methodology can optimise the power and energy consumption for a reliability constrained processor system. Lowering the supply voltage of processor core and L1-cache reduces both the power consumption and reliability. L1-cache resizing is proposed to mitigate the impact of VFS on L1-cache reliability and dynamic ECC protection is used on L2-cache and DRAM to further reduce energy consumption. Table 5.5 shows under the reliability constraint of $2.5 \times 10^5 \lambda_{bit}$, the energy saving achieved by using both L1-cache resizing and dynamic protection of L2-cache and DRAM. The first column shows the benchmark application; the second main column shows minimum energy consumption under VFS and its corresponding supply voltage which was limited by reliability constraints; the third main column shows the minimum energy consumption under VFS when L1-cache resizing and dynamic ECC protection are used

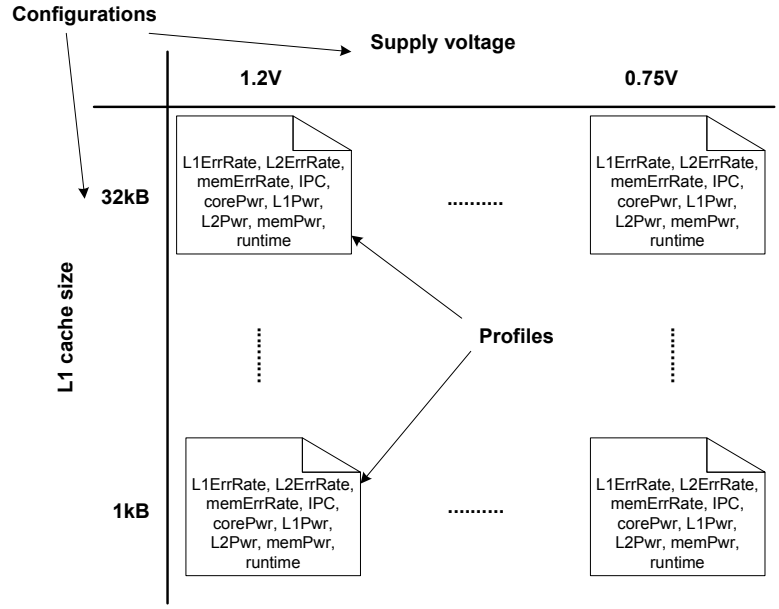


Figure 5.31: Example of Reliability, Power and Performance Profiles (RPPP).

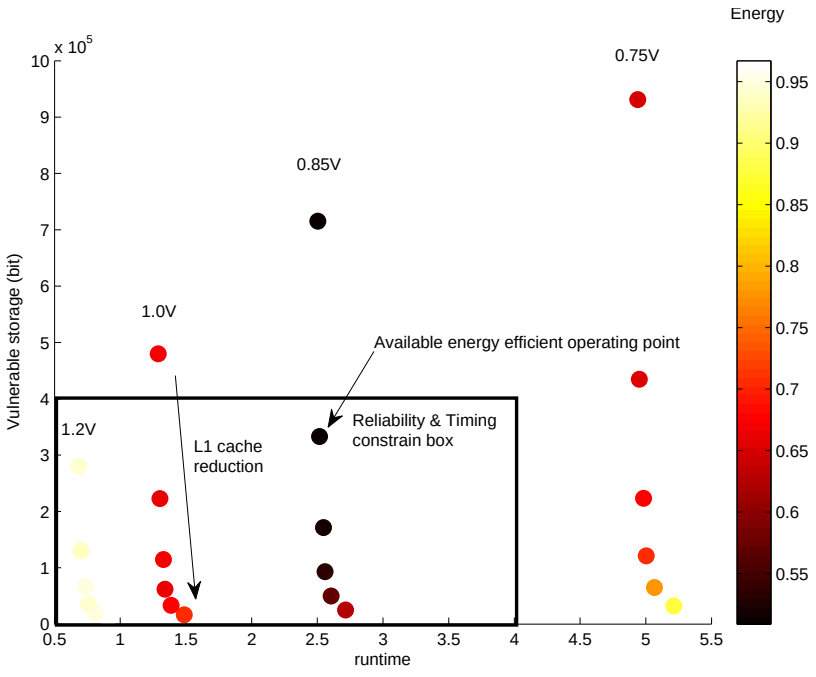


Figure 5.32: Example of reliability and performance constrained energy optimisation for application "lame".

Table 5.5: Energy saving by combining VFS, L1-cache resizing and dynamic protection under memory components reliability constraint of $2.5 \times 10^5 \lambda_{bit}$, the constraint is chosen for demonstration purposes

	VFS		VFS, L1-cache resizing, dynamic protection					
	Vdd	Energy	Vdd	L1	ECC		Energy	
Application	(V)	(J)	(V)	(kB)	L2	DRAM	(J)	saving
basicmath	0.85v	1.696	0.85v	32kB	0	0	1.568	7.5%
bitcount	0.85v	0.376	0.85v	1kB	0	0	0.342	9.0%
qsort	1.2v	0.292	0.85v	16kB	1	1	0.202	31.1%
susan_corners	1.2v	0.020	0.85v	4kB	1	1	0.013	34.5%
susan_edges	1v	0.039	0.85v	8kB	1	1	0.035	10.4%
susan_smoothing	1.2v	0.229	0.85v	8kB	1	0	0.135	40.9%
djpeg	1v	0.015	0.85v	16kB	1	0	0.012	19.0%
cjpeg	0.85v	0.056	0.85v	32kB	1	1	0.056	0.0%
lame	1.2v	0.951	0.85v	16kB	1	1	0.648	31.9%
typeset	1.2v	0.421	0.85v	16kB	1	1	0.345	18.1%
dijkstra	1.2v	0.159	0.85v	8kB	1	0	0.111	30.4%
patricia	1.2v	0.596	0.85v	16kB	1	1	0.474	20.4%
ispell	1.2v	0.803	0.85v	16kB	1	1	0.606	24.6%
rsynth	1.2v	1.696	0.85v	8kB	1	1	1.252	26.2%
stringsearch	0.85v	0.003	0.85v	16kB	0	0	0.003	7.1%
rijndael_enc	1v	0.281	0.85v	16kB	0	0	0.230	18.3%
rijndael_dec	1v	0.271	0.85v	8kB	0	0	0.232	14.3%
sha	0.85v	0.063	0.85v	2kB	0	0	0.058	8.3%
crc	1v	1.580	1v	2kB	0	0	1.445	8.5%
fft	1.2v	0.421	0.85v	16kB	1	1	0.311	26.2%
ifft	1.2v	0.232	0.85v	16kB	1	1	0.173	25.6%
cadpcm	0.85v	0.362	0.85v	4kB	0	0	0.332	8.1%
dadpcm	0.85v	0.259	0.85v	4kB	0	0	0.239	8.0%
gsm_toast	1.2v	0.880	0.85v	16kB	0	0	0.588	33.2%

to improve the reliability of cache memory which allows the processor to operate under a more energy efficient supply voltage point; and the last column shows the energy saving in percentage terms. For some applications, supply voltage scaling is limited by the reliability constraints of L1-cache, which is shown in the second column. L1-cache resizing mitigates the impact of VFS on reliability; thus for all applications the supply voltage for minimum energy consumption can be used as shown in the fourth column. The 6th and 7th columns show the enable signals for ECC protection of L2-cache and DRAM: '1' means enabling and '0' means disabling protection. It shows that L2 cache and DRAM ECC protection are only enabled in the applications where their reliability is lower than $2.5 \times 10^5 \lambda_{bit}$. Comparing to the processor system with only VFS, the proposed methodology achieves energy saving for most applications under reliability constraint. The average saving is 19% and the maximum saving is 40% .

5.5 Concluding Remarks

The reliability of memory systems is critical for the correct operation of processors as memory occupies most of the chip area, which is increasingly susceptible to soft errors as transistor size scaling reduces critical charge and increases device density. Furthermore low power techniques such as voltage and frequency scaling exacerbate the problems. ECC protection requires hardware redundancy and consumes energy, and blanket protection can be costly especially for energy sensitive designs. Therefore accurate estimation of memory reliability with targeted protection is needed.

This chapter demonstrates an analysis framework that supports co-optimisation of processor's performance, energy and reliability. Through the analysis of memory system reliability, it was shown that the L1-cache is the most reliable memory component follow by the L2 cache, and DRAM is the least reliable memory component. A cost effective reliable design methodology is proposed to help hardware redundancy insertion and dynamic ECC protection against soft error. The impact of voltage scaling on memory system reliability has also been studied. L1-cache resizing was proposed to mitigate the impact of voltage and frequency scaling on memory reliability, which itself has little impact on performance and energy consumption under reduced supply voltage. A reliability aware joint optimisation flow utilizing both dynamic ECC protection and L1-cache resizing has been proposed for energy minimisation. Finally, it has been demonstrated through joint optimisation examples that, energy reduction can be achieved for most applications in a reliability-constrained design.

Chapter 6

Conclusion and Future Work

With technology scaling, leakage and dynamic power reduction techniques continue to evolve as power minimization is an essential part of modern electronic system design. Power gating and voltage scaling are two popular power reduction techniques, which are effective in reducing power consumption but exacerbate reliability problems. The contributions presented in this thesis provided novel and cost effective techniques to improve the reliability of storage elements of modern low power electronic systems, including processor registers and memory systems, which not only constitute up to 70% of total silicon area but play a central role in reducing power consumption. These techniques have been implemented using standard EDA tools and validated on commercial low power processors. The contributions are summarised in the next section followed by proposed future work.

6.1 Thesis Contributions

All the objectives of this thesis are listed in Chapter 2, Section 2.6. The first objective is achieved by studying register designs and technology:

1. An in-depth analysis of critical charge of flip-flop register storage node was given in Chapter 2, Section 2.3. The effect of supply voltage scaling on critical charge of flip-flops in the presence of process variation was investigated through SPICE simulations using 45-nm CMOS technology with 0.9-V nominal supply voltage setting. It was found that the critical charge of the flip-flops reduces with the reduction in supply voltage and it further deteriorates due to the effect of process variation. These results demonstrated that the critical charge of flip-flops is negatively affected not only because of reduction in supply voltage but also due to process variation. Therefore supply voltage scaling and process variation can significantly increase transient error rate.

2. Chapter 4, Section 4.1 presented an analysis based on measurement results from a total of 82 test chips, which were used to characterise state integrity challenges due to voltage and temperature variations, for the baseline silicon process and its inherent variability. It was shown that even at 25°C, the state integrity of flip-flops is affected by process variation leading to spread measurement of First Failure Voltage (FFV), from 245-mV to 315-mV, with 79% of total dies exhibiting single bit failure at FFV, while the rest showing multi-bit failure at FFV. Furthermore, at elevated temperatures the variation is even more pronounced. It was found that FFV increases by up to 30-mV with increase in temperature from 25°C to 79°C. The effect of process variation with geometry scaling has also been studied using a 45-nm technology node through Monte-Carlo simulation; when compared with 65-nm technology, it was found that the overall distribution trend remained the same, however the mean FFV moved to a higher voltage. An important observation from these results was that using a fixed state retention voltage across all dies is sub-optimal because it fails to minimise the leakage power of all dies during sleep mode.

The second objective is achieved by the following:

1. Chapter 3, Section 3.1 presented a technique to mitigate the impact of power gating and supply voltage scaling on the reliability of embedded processors. The proposed technique is effective because it uses two methods of error recovery. The first method of error recovery is through hardware implementation of Hamming code that is capable of 1-bit error correction per codeword; to recover multi-bit (clustered) errors, it employs hardware implementation of CRC for error detection and software state recovery to restore the last known good states of the design. The proposed technique is low-cost because it reuses scan chains that are used in manufacturing test and it is energy efficient because it partitions scan chains to minimise the time needed for state monitoring and recovery. It was shown that the hardware-based Hamming error correction improved system reliability by more than 2 order of magnitude compared to a design without error protection, when considering maximum soft error rate observed at normal operating conditions ($<= 10^{?7}$ errors per bit-hour). The system failure rate increases significantly at higher bit error rate ($> 10^{?7}$ errors per bit-hour) and in the case of clustered errors. For such a high failure rate, a software state recovery method is used for efficient state recovery.

2. An important finding from the analysis in Chapter 4, Section 4.1 is that using a fixed state retention voltage across all dies is sub-optimal because it fails to minimise the leakage power of all dies during sleep mode. To minimise leakage power while ensuring state integrity, minimum retention voltage (MRV) of each individual die should be characterised. Chapter 4 Section 4.2 presented an effective

technique to improve the minimum retention power and state-integrity of voltage scaled flip-flops under process, voltage and temperature variation. The proposed technique consists of the following two steps. Firstly, a characterisation algorithm is used to determine MRV of a given die, as it varies due to process variation. The characterisation step of a die is an offline process and is performed only once per die. Secondly, a control flow for error detection and single-bit error correction is proposed, which relies on horizontal and vertical parity; whenever an error is detected, it raises the characterised minimum retention voltage to reduce subsequent error possibility. The prototype of the proposed control flow is implemented in a host computer using a script written in Python, which provides voltage scaling by controlling the test chip through an external power supply. Silicon results show that at characterised MRV, the flip-flop state integrity is preserved, while achieving up to 17.6% reduction in retention voltage across 82-dies.

3. The reliability of memory system hierarchy is critical for the correct operation of processors as memory occupies most of the chip area, which is increasingly susceptible to soft errors as transistor size scaling reduces critical charge and increases device density. Furthermore, low power techniques such as voltage and frequency scaling exacerbate the problem by reducing critical charge. ECC protection requires hardware redundancy and consumes energy, and blanket protection can be costly particularly for energy conscious designs. Therefore accurate estimation of memory reliability with targeted protection is needed. Chapter 5 described an architectural simulation-based framework, which enables joint analysis of reliability, performance and energy consumption of embedded processor memory systems. Through this analysis, a cost effective reliable design methodology is proposed to guide hardware redundancy insertion and dynamic ECC protection against soft error. Level 1 cache resizing is proposed to mitigate the impact of voltage and frequency scaling on reliability, which itself has little impact on performance and energy consumption under reduced supply voltage. A joint flow optimisation approach to reliability aware is proposed for energy minimization which utilizes both dynamic ECC protection and L1 cache resizing. It is demonstrated through the joint optimisation examples that energy reduction can be achieved for most applications in a reliability constrained design.

The third objective is achieved by the following:

1. The technique presented in Chapter 3, Section 3.1 was validated through two case studies. The first case study used FPGA implementation of a register-rich FIFO design and it was shown that hardware based implementation of Hamming code is capable of effective multiple bit error detection and single bit error recovery. The second case study implemented the complete solution (Hardware and Software based recovery) applied to an industry standard embedded microprocessor ARM

Cortex-M0, and it was validated on an FPGA and further synthesised using 65-nm technology library and Synopsys EDA tool.

2. Using 65-nm test chip implementation, Chapter 4, Section 4.4 described two experiments conducted to demonstrate improved state integrity of flip-flops with aggressive supply voltage scaling that is possible through the technique presented in Chapter 4, Section 4.2. The first experiment demonstrates improved state integrity of flip-flops in "sleep state", and the second experiment demonstrates the effect of aggressive supply voltage scaling on leakage power savings. Silicon results show that at characterised MRV, the flip-flop state integrity is preserved, while achieving up to 17.6% reduction in retention voltage across 82-dies.

The final objective is achieved by the following:

1. The technique presented in Chapter 3, Section 3.1 can be incorporated into the existing power gating and voltage scaling design synthesis flow which was described in Chapter 3, Sections 3.1.2 and 3.1.3.

2. The technique presented in Chapter 4, Section 4.2 can be incorporated in conventional design synthesis flow. Chapter 4, Section 4.2.3 described an additional step needed to automate the insertion of horizontal and vertical parity logics

These three contributions presented in this thesis provide novel, relevant and cost-effective reliability solutions for different types of storage elements that are widely used in low power designs including register files and memory systems. The conclusions drawn in this thesis are supported by extensive analysis using state-of-the-art EDA tools, in-house software specifically developed to generate realistic data and workloads, FPGA implementation of commercial low power embedded processors and measured results from fabricated chip designs to meet the last objective of this thesis. It is hoped that the low cost and effective reliability solutions proposed in this thesis will make useful contributions towards the development of future low-power design methods and EDA tools.

6.2 Future Work Directions

Based on the research presented in this thesis, a number of directions for future studies have been identified and are outlined in the following:

6.2.1 Improving Reliability of Power Management Hardware

In Chapter 3, the concept of on-chip reuse is introduced to provide resilience to energy-efficient embedded processors at low-cost (area, energy) and with little impact on functional performance. Further research could be conducted to investigate and develop a low-cost, on-line soft-error monitoring and correction method and its associated implementation circuitry for power management hardware. Power controllers, state retention registers and isolation cells are more vulnerable to soft errors than other power management hardware and therefore more emphasis will be given to these power management components. Currently, there are no reported on-line soft-error monitoring and correction methods for power management hardware; making it impossible to isolate and correct errors and leaving them vulnerable during operation. Special consideration will be given to the concept of on-chip reuse to reduce the implementation cost (area and energy) of on-line soft error monitoring. A good starting point to achieve this is reusing the circuitry of built-in self-test (traditionally used for permanent manufacturing defects testing) consisting of pattern generators and comparators for on-line test error-detection. In this way it may be possible to lower the implementation cost of protecting the power management hardware from soft errors. For error correction, the use of error correction codes and software state recovery will be investigated, thereby providing an effective and low-cost on-line monitoring and correction method. The contributions described in Chapter 3 and Chapter 4 of this thesis will be the starting point for this work.

6.2.2 Reliability Enhancement of Multi-core Processors through Hardware-Software Co-design

Balancing trade-offs between performance, energy consumption and reliability is essential for embedded processors. In Chapter 5, a joint optimisation technique was proposed to minimise energy consumption while meeting reliability constraint for a single-core embedded processor. For multi-core embedded processors, energy minimization can be achieved through careful scheduling, whilst reliability can be improved through monitoring and targeted protection. By combining scheduling, reliability monitoring and targeted protection, further energy reduction can be achieved within the performance and reliability constraints. Analysis framework for multi-core processors similar to the one reported in Chapter 5 is required for hardware-software design space exploration and investigating various scheduling algorithms and reliability monitoring techniques.

This thesis focused on improving reliability of processors against transient errors. Permanent faults due to various ageing effects as shown in Figure 6.1 can cause digital circuits to malfunction prematurely. In addition to soft errors, further research could be conducted to develop techniques for predicting and detecting faults occurrence due to ageing effect with the aim of slowing down the ageing process; as an example, placing

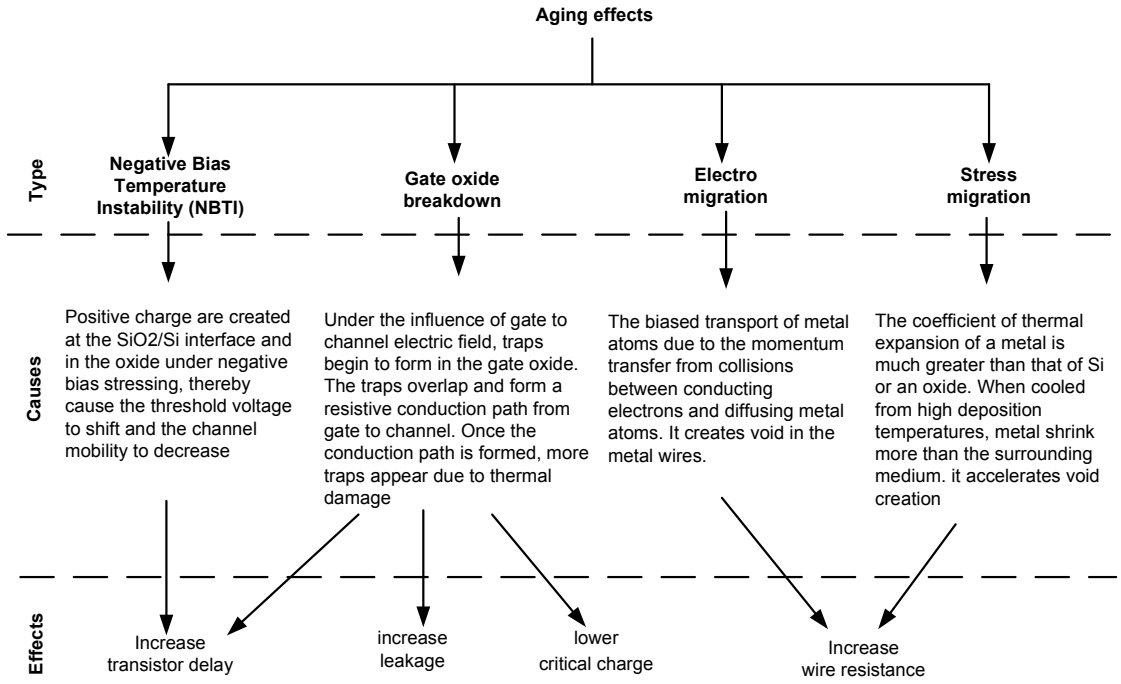


Figure 6.1: Various Aging Effects

processor cores in recovery mode when they are not in use and balancing work-loads across various processor cores. To achieve this, statistical analysis, similar to the one reported in Chapter 5 Section 5.3 is required to understand the work-load distribution and low cost circuitry or design techniques that are required for work-load balancing.

Appendix A

Low Power Embedded Processors

This appendix gives an overview of the ARM Cortex-M0 microprocessor used in Chapter 3 and 4. In Chapter 3, it was used as a case study to validate the proposed state monitoring and recovery technique. In Chapter 4, it was implemented in test chips to provide state monitoring for PVT aware state protection techniques. The content of this section is gathered from the ARM Cortex-M0 Technical Reference Manual [187].

Embedded processors are designed to perform some specific functions and it is embedded in a integrated system that often includes electronics and mechanical parts. Low power and high energy efficiency are the key design parameters for embedded processors. In lower power embedded processors, processors can be powered off when it is idle. ARM Cortex-M0 (CM0) is a modern energy-efficient ARM microprocessor, which is used in many low power embedded applications such as medical devices, e-metering, power and motor control [150]. The processor uses Thumb-2 instruction set, which is the extension of 16-bit Thumb instruction by adding some 32-bit instructions to achieve balance between code density and performance. It achieve energy efficiency while improve performance over 8-bit micro-controllers, the throughput is 3x better than the Texas Instruments MSP430 and 2x better than the Microchip PIC24 [188].

Figure A.1 shows the block diagram of CM0 processor [187]. The essential units of the processor are processor core, Nested Vector Interrupt Controller (NVIC) and bus matrix. The processor core is based on the ARMv6-M^{TM} architecture with 3 stages pipeline including instruction fetch, instruction decode and excution. The processor core consist of a register bank, a ALU, a datapath and control logics. In the register bank, there are 12 general purpose registers, a stack pointer, a link register and a program counter. In addition, there are special purpose registers: program status register contains the programs information in terms of application, interrupt and execution. Control registers controls the stack used when processor is in thread mode. The ALU is made of integer adder and optional single cycle hardware multiplier (without it multiplication is done using software emulation). The NVIC can be configured to handle upto 32 interrupt requests with configurable priority system, in addition to a non-maskable NMI interrupt. The purpose of NMI interrupt controller is to handle nested interrupts and

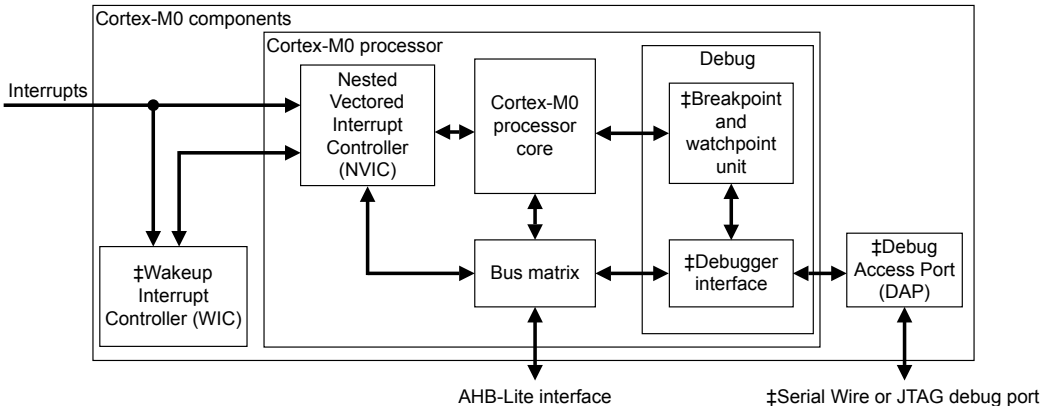


Figure A.1: ARM Cortex-M0 processor block diagram [187].

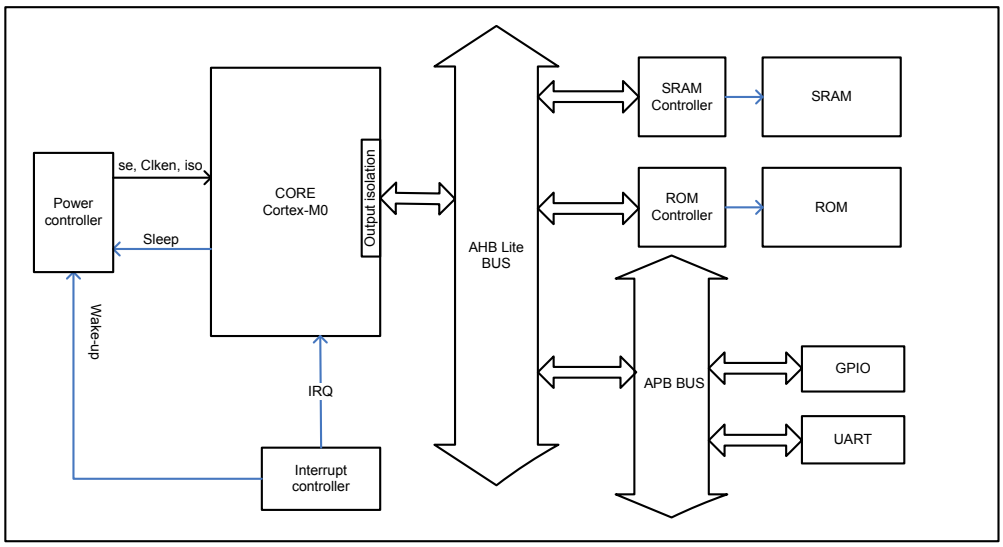


Figure A.2: ARM Cortex-M0 based embedded processor system block diagram.

ensure the correct excutions of Interrupt Searvice Routines (ISR). In this work, this interrupt controller is used to handle the wake-up request (Chatper 3 Section 3.3) and error detection event (Chatper 4 Section 4.2.2). CM0 uses Von Neumann architecture, therefore the bus matrix has a unified 32-bit bus for both instruction and data. The bus interface uses Advanced Microcontroller Bus Architecture AHB-Lite specification.

Figure A.2 shows the example implementation a lower power embedded processor system based on the ARM Cortex-M0 processor. The processor core is connected to both ROM and SRAM controller through the AHB-Lite bus. In lower power embedded processors system, processor core can be powered off when it is idle. The power controller and interrupt controller are used to control the power cycles of the processor core. The General Purpose Input and Output (GPIO) and the Universal Asynchronous Receiver and Transmitter (UART) are connected to processor core through the APB bus to provide communication to workstation.

Appendix B

Embedded Processor Power Domain Description

This appendix describes the power domain arrangement for Cortex-M0 low power embedded processor implementation in Chapter 3, whose graphical representation is shown in Chapter 3, Figure 3.19. The power domain is described in universal power format (UPF). Three power domains are created: *TOP_PD*, *CORE_PD* and *SMRB_PD*. Processor core is located in *CORE_PD*, which can be power down during sleep mode. State Monitoring and Recovery Block (SMRB) is located in *SMRB_PD*, which is only powered when encoding before sleep mode and decoding after sleep mode. The rest of processor system is placed in *TOP_PD* that is kept always-on. The UPF description is divided into the following parts:

1. Power Domains: creates power domains.
2. Power Supply Nets: creates power supply nets.
3. Assign Primary Power Supply Nets to Power Domains.
4. Power Switches: creates power switches and connecting them to supply rails and control signals.
5. Isolation Strategy: creates signal isolations across power domains to prevent the prorogation of floating signals from the power downed block.
6. State Retention: Instructs synthesis tool to create state retention enabled flip-flops, and connecting them to state preserving supply rails and control signals

```
#-------------------------------------------------------------
# System Power Domain Description Using UPF
#-------------------------------------------------------------

# Part 1: Power Domains

create_power_domain TOP_PD
create_power_domain CORE_PD -elements {MINISWIFT}
create_power_domain SMRB_PD -elements {SMRB}

# Part 2: Power Supply Nets

create_supply_port PAD_VDD
create_supply_net VDD -domain TOP_PD
create_supply_net VDD -domain CORE_PD -reuse
connect_supply_net VDD -ports PAD_VDD

create_supply_port PAD_VSS
create_supply_net VSS -domain TOP_PD
create_supply_net VSS -domain CORE_PD -reuse
connect_supply_net VSS -ports PAD_VSS

create_supply_net CORE_VDD -domain CORE_PD
create_supply_net SMRB_VDD -domain SMRB_PD

# Part 3: Assign Primary Power Supply Nets to Power Domains

set_domain_supply_net TOP_PD \
    -primary_power_net VDD \
    -primary_ground_net VSS

set_domain_supply_net CORE_PD \
    -primary_power_net CORE_VDD \
    -primary_ground_net VSS

set_domain_supply_net SMRB_PD \
    -primary_power_net SMRB_VDD \
    -primary_ground_net VSS

# Part 4: Power Switches
```

```
create_power_switch core_switch -domain CORE_PD \
    -input_supply_port {VDD VDD} \
    -output_supply_port {CORE_VDD CORE_VDD} \
    -control_port {power pg_ctrl/core_power} \
    -on_state {on_state VDD {core_power}} \
    -off_state {off_state {!core_power}}

create_power_switch smrb_switch -domain SMRB_PD \
    -input_supply_port {VDD VDD} \
    -output_supply_port {SMRB_VDD SMRB_VDD} \
    -control_port {power pg_ctrl/smrb_power} \
    -on_state {on_state VDD {smrb_power}} \
    -off_state {off_state {!smrb_power}}

# Part 5: Isolation Strategy

set_isolation core_iso -domain CORE_PD \
    -isolation_power_net VDD \
    -isolation_ground_net VSS \
    -clamp_value 0 \
    -applies_to outputs

set_isolation_control core_iso -domain CORE_PD \
    -isolation_signal pg_ctrl/core_nclamp \
    -isolation_sense low \
    -location self

set_isolation smrb_iso -domain SMRB_PD \
    -isolation_power_net VDD \
    -isolation_ground_net VSS \
    -clamp_value 0 \
    -applies_to outputs

set_isolation_control smrb_iso -domain SMRB_PD \
    -isolation_signal pg_ctrl/smrb_nclamp \
    -isolation_sense low \
    -location self

# Part 6: State Retention

set_retention core_ret -domain CORE_PD \
```

```
    -retention_power_net VDD \
    -retention_ground_net VSS

set_retention_control core_ret -domain CORE_PD \
    -save_signal {pg_ctrl/core_retain high} \
    -restore_signal {pg_ctrl/core_retain low}

set_retention smrb_ret -domain SMRB_PD \
    -retention_power_net VDD \
    -retention_ground_net VSS

set_retention_control smrb_ret -domain SMRB_PD \
    -save_signal {pg_ctrl/smrb_retain high} \
    -restore_signal {pg_ctrl/smrb_retain low}
```

Appendix C

Firmware For Software Recovery

This appendix describes the software recovery firmware used in Chapter 3, Section **??**. There are two main functions: Function *save_state()* is used to save architectural states before going to sleep mode, and Function *restore_state()* is used to restore the original architectural states if state corruption is detected during sleep mode. The test function *main()* is an infinite loop with the following steps:

1. Processor operates in normal functional mode.
2. Before going to sleep, save architectural states.
3. Function *__WFI()* puts processor into sleep mode and waiting for interrupts.
4. CRC is used to check the state integrity of the processor upon receiving an interrupt and before waking up the processor.
5. If errors are detected, reset the processor invoking initiation function *Reset_Handler()*.
6. Fuction *Reset_Handler()* first checks what caused the reset. If the reason is state corruption, it calls function *restore_state()* to restore architectural states.

Step 4 is carried out in hardware, which is not shown in this appendix.

```
//-------------------------------------------------------------
// Processor architectural states saving and restoring firmware
//-------------------------------------------------------------

//main function
int main(){
    while(1){
        //do some thing ...........
```

```
        //save architectural states into memory before going to sleep
        save_state();
        __WFI(); //put processor into sleep and waiting for interrupt
    }
}

//function for saving the status and control registers of processor core
void save_state(void){
    save_reg[0x0] = NVIC->ISER[0];
    save_reg[0x1] = NVIC->ISPR[0];
    save_reg[0x2] = NVIC->IP[0];
    save_reg[0x3] = NVIC->IP[1];
    save_reg[0x4] = NVIC->IP[2];
    save_reg[0x5] = NVIC->IP[3];
    save_reg[0x6] = SCB->ICSR;
    save_reg[0x7] = SCB->AIRCR;
    save_reg[0x8] = SCB->SCR;
    save_reg[0x9] = SCB->CCR;
    save_reg[0xa] = SCB->SHP[0];
    save_reg[0xb] = SCB->SHP[1];
    save_reg[0xc] = SysTick->CTRL;
    save_reg[0xd] = SysTick->LOAD;
    save_reg[0xe] = SysTick->VAL;
    save_regfile(&save_reg[0xf]);
}

// assembly function for saving register file
// register r0 stores the input parameter of the function
__ASM void save_regfile(uint32_t *preg){
    //save r1-r3
    STM r0!,{r1-r3}
    //get r4 from stack
    LDR r4,[sp,#0x0]
    //save r4-r7
    STM r0!,{r4-r7}
    //save r8-r12, primask, control
    //the storing instruction can only access r0-r7
    //other registers value needs to be moved to r1-r7 first
    MOV r1,r8
    MOV r2,r9
    MOV r3,r10
```

```
    MOV r4,r11
    MOV r5,r12
    MRS r6,primask
    MRS r7,control
    STM r0!,{r1-r7}
    MRS r1,psr
    MRS r2,psp
    MRS r3,msp
    //get link register value from the stack and keep in r4
    //the link register stored the current value of the this function,
    //the previous link register was moved to stack
    LDR r4,[sp,#0x4]
    STM r0!,{r1-r4}
    //restore r1-r7
    MOVS r1,#0x48
    SUBS r0,r0,r1
    LDM r0!,{r1-r7}
    BX  lr
}

//restor states if the reset is caused by error event
void Reset_Handler(void)
{
    if(STAT_REG->ERR_DETECT_HALFWORD != 0x0000){
        increase_stack(8);
        restore_state();
    }
    __main();
}

__ASM void increase_stack(uint32_t offset){
    MRS r1,msp
    ADDS r1,r0
    MSR msp,r1
    BX lr
}

//function to restore private registers of processor core
void restore_state(void){
    NVIC->I
    SER[0]  =  save_reg[0x0];
```

```
    NVIC->ISPR[0]  = save_reg[0x1];
    NVIC->IP[0]  = save_reg[0x2];
    NVIC->IP[1]  = save_reg[0x3];
    NVIC->IP[2]  = save_reg[0x4];
    NVIC->IP[3]  = save_reg[0x5];
    SCB->ICSR  = save_reg[0x6];
    SCB->AIRCR  = save_reg[0x7];
    SCB->SCR  = save_reg[0x8];
    SCB->CCR  = save_reg[0x9];
    SCB->SHP[0]  = save_reg[0xa];
    SCB->SHP[1]  = save_reg[0xb];
    SysTick->CTRL  = save_reg[0xc];
    SysTick->LOAD  = save_reg[0xd];
    SysTick->VAL  = save_reg[0xe];
    restore_regfile(&save_reg[0xf]);
    STAT_REG->ERR_DETECT_HALFWORD = 0x0000;
}

//assembly function to restore register file
__ASM void restore_regfile(uint32_t *preg){
    //restore r8-r12,primask,control
    //instruction can only access r0-r7,
    //so other registers values are load into r1-r7 first
    MOVS r1,#0x1c
    ADDS r0,r0,r1
    LDM r0!,{r1-r7}
    MOV r8,r1
    MOV r9,r2
    MOV r10,r3
    MOV r11,r4
    MOV r12,r5
    MSR primask,r6
    MSR control,r7
    //restore psr,psp,msp,lr
    LDM r0!,{r1-r4}
    //move stack information to recovered stack top
    //get the value of original r4 and keep in r6
    MOVS r5,#0x48
    SUBS r0,r0,r5
    LDR r6,[r5,#0xc]
    //increase original lr by 2
```

```
    MOVS r7,#0x2
    ADDS r4,r4,r7
    MSR psr,r1
    MSR psp,r2
    //restore main stack pointer
    MSR msp,r3
    //save r4 and modified lr in restored stack
    STR r6,[sp,#0x0]
    STR r4,[sp,#0x4]
    //restore r1-r7
    LDM r0!,{r1-r7}
    BX lr
}
```

Appendix D

HSPICE Monte-Carlo Simulation

This appendix describes the HSPICE scripts for Monte-Carlo simulation searching boundary condition of 1-to-0 bit-flip. This was used to generate the simulation results presented in Chapter 4 Figure 4.4.(b) for First Failure Voltage (FFV) analysis. Figure 4.4.(c) was generated using a similar script, the only difference is in standard cell library. The change needed to simulate 0-to-1 bit-flip is marked in the script. The scripts is divided into the following parts:

1. Set supply voltage, retention voltage and retention time.
2. Instruct HSPICE to use Gaussian distribution for Monte-Carlo simulation.
3. Use HSPICE optimisation tool to search the boundary condition where a flip-flop can barely retain its state.
4. Run transient Monte-Carlo simulation with 8192x82=671744 iterations emulating 8192 flip-flops in 82 chips.
5. Create stimulus to store 1 into flip-flop and switch between supply voltage and retention voltage
6. Modify original standard cell description to enable the variation of transistor parameters.

```
*-------------------------------------------------------------
* HSPICE Monte-Carlo simulation for searching
* First Failure Voltage (FFV) of 1-to-0 bit-flip
*-------------------------------------------------------------

*Setting parameters
*Part 1: Set supply voltage, retention voltage and retention time
.PARAM Supply=1.2
.PARAM RetSupply= Opt1 (0, 0, 'Supply*0.9')
.PARAM RetTime=300us
.PARAM RunTime='RetTime+10us'
.PARAM rt=10P
.PARAM ft=10P

*Part 2: Instruct HSPICE to use Gaussian distribution
*        for Monte-Carlo simulation
*these parameters will be used in standard cell library
*to emulate transistor process variation
.PARAM monte_l=AGAUSS(0,7.2n,3)
.PARAM monte_vth0=AGAUSS(0,0.023,3)
.PARAM monte_u=AGAUSS(1,0.63,3)
.PARAM monte_tox=AGAUSS(0,0.06n,3)

*Part 3: Use HSPICE optimisation tool to search the boundary
*        condition where a flip-flop can barely retain its state
.Model OptMod OPT METHOD=PassFail
.Meas Tran Tprop Trig v(VDD) Val=Supply Rise=1 TD=10ns
+ Targ v(Q) Val=Supply Rise=1

.Model OptMod OPT METHOD=PassFail
.Meas Tran Tprop Trig v(VDD) Val=Supply Rise=1 TD=10ns
+                Targ v(Q) Val=Supply Rise=1


.Option NoMod AutoStop
.Option Seed=random
*Part 4: Run transient Monte-Carlo simulation with 8192x82=671744
*        iterations emulating 8192 flip-flops in 82 chips
.TRAN 0.1n RunTime Sweep Optimise=Opt1 Results=Tprop
+ Model=Optmod Monte=671744
.TEMP 25
```

```
*transistor models
.lib '/home/syang/work/HSPICE/lib/CLN65LP_2d5_lk_v1d3.l' TT

*65lp spice subckt files
.INCLUDE '/home/syang/work/HSPICE/lib/MDFFHQ_X1_A8TR_monte.sp'

*Part 5: Create stimulus to store 1 into flip-flop and
*        switch between supply voltage and retention voltage
VDD VDD 0 PWL (0 Supply 3ns Supply '3ns+ft' RetSupply '3ns+ft+RetTime'
+ RetSupply '3ns+ft+RetTime+rt' Supply)
VCK  CK  0 PWL (0 0 1ns 0 '1ns+rt' Supply '2ns+rt' Supply '2ns+rt+ft' 0)
*store '1' to flip-flop
VD0  D0  0 PWL (0 Supply 1.5ns Supply '1.5ns+ft' 0)
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*change required for 0-to-1 bit-flip experiment
*store '0' to flip-flop
*VD0  D0  0 PWL (0 0 3ns 0 '3ns+ft' RetSupply '3ns+ft+RetTime' RetSupply
*+ '3ns+ft+RetTime+rt' Supply)
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Cload Q 0 0.01p


*Device under test
XMDFF Q VDD VDD 0 0 CK D0 0 0 MDFFHQ_X1_A8TR
.END

*-----------------------------------------------------------------------
* Part of standard cell HSPICE description file MDFFHQ_X1_A8TR_monte.sp
* modified for Monte-Carlo simulation
*-----------------------------------------------------------------------

*Part 6: Modify original standard cell description
*        to enable the variation of transistor parameters
MMX_T0 BM:F129 C:F130 M:F131 VPW nch_hvt ad=0.038125p as=0.03433p
+ nrd=0.027322 nrs=0.021673 pd=0.6875u ps=0.526786u sa=0.883822u
+ sb=0.313568u sca=16.1064 scb=0.017629 scc=0.001687 w=0.25u
+ l='0.060168u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MMX_T1 M:F205 CN:F206 BM:F207 VNW pch_hvt ad=0.102p as=0.066764p
+ nrd=0.175 nrs=0.103299 pd=1.42u ps=1.25182u sa=0.2u sb=0.160823u
```

```
+ sca=10.0365 scb=0.010112 scc=0.000839 w=0.51u
+ l='0.060022u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MXI12/MXNA1 VSS:F139 S:F138 XI12/N1:F137 VPW nch_hvt ad=0.024p
+ as=0.010125p nrd=0.541667 nrs=36.1636 pd=0.62u ps=0.285u
+ sa=1.81441u sb=0.16u sca=17.6981 scb=0.020292 scc=0.001923 w=0.157222u
+ l='0.060053u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MXI12/MXNOE BM:F133 CN:F134 XI12/N1:F135 VPW nch_hvt ad=0.022875p
+ as=0.010125p nrd=0.006148 nrs=36.1636 pd=0.4125u ps=0.285u
+ sa=1.59323u sb=0.355u sca=17.6854 scb=0.020292 scc=0.001923 w=0.150085u
+ l='0.06u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MXI12/MXPA1 VDD:F215 S:F214 XI12/P1:F213 VNW pch_hvt ad=0.0255p
+ as=0.01275p nrd=0.509804 nrs=27.2919 pd=0.64u ps=0.32u sa=0.79u sb=0.17u
+  sca=4.73771 scb=0.003293 scc=2.5e-05 w=0.153294u
+ l='0.06u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MXI12/MXPOEN BM:F209 C:F210 XI12/P1:F211 VNW pch_hvt ad=0.019636p
+ as=0.01275p nrd=0.00434 nrs=27.2919 pd=0.368182u ps=0.32u sa=0.56u
+ sb=0.4u sca=4.71936 scb=0.003293 scc=2.5e-05 w=0.15u
+ l='0.06u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MX_G2/MXNA1 S:F145 BM:F146 VSS:F147 VPW nch_hvt ad=0.02625p as=0.015886p
+ nrd=0.028571 nrs=0.01073 pd=0.65u ps=0.354545u sa=0.175u
+ sb=0.43u sca=11.0038 scb=0.013065 scc=0.000623 w=0.151381u
+ l='0.06u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox

MX_G2/MXPA1 S:F217 BM:F218 VDD:F219 VNW pch_hvt ad=0.02625p as=0.022821p
+ nrd=0.028571 nrs=0.004401 pd=0.65u ps=0.391071u sa=0.175u
+ sb=0.195258u sca=20.7887 scb=0.022682 scc=0.00261 w=0.155118u
+ l='0.06u+monte_l' delvt0=monte_vth0 mulu0=monte_u deltox=monte_tox
```

Appendix E

Example Script for Joint Optimisation

This appendix describes the example script of joint optimisation algorithm written in Python, which was used to minimise energy consumption under performance and reliability constraints for processor memory system. The corresponding algorithm was described in Chapter 3 Figure 5.30. The script is divided into the following parts:

1. Fetch Reliability, Power and Performance Profiles (RPPP) from pre-characterised optimisation table.
2. Filter RPPP by performance and reliability constraints.
3. looking for a RPPP with minimum energy.

```
#-------------------------------------------------------
# Joint optimisation script written in Python
#-------------------------------------------------------

def joinOpt():
  infile = open('joinOptTable.dat', 'rb')
  joinOptTable = pickle.load(infile)
  infile.close()
  l2EccCost = 1.25
  memEccCost = 1.5
  maxErrRate = 260000
  maxRuntime = 'inf'
  maxErrRate = maxErrRateFloor
  for app in benchmarks:
    maxErrRate = maxErrRateFloor
    minEnergy = float('inf')
```

```
minPower = float('inf')
minEnergyNoResize = float('inf')
minPowerNoResize = float('inf')
for vol in voltages:
  for csize in caches:
      #Part 1: Fetch RPPP
      pwrCore = joinOptTable[app][vol][csize]['pwrCore']
      pwrL2 = joinOptTable[app][vol][csize]['pwrL2']
      pwrMem = joinOptTable[app][vol][csize]['pwrMem']
      l1SER = joinOptTable[app][vol][csize]['l1SER']
      l1iSER = joinOptTable[app][vol][csize]['l1iSER']
      l1dSER = joinOptTable[app][vol][csize]['l1dSER']
      l2SER = joinOptTable[app][vol][csize]['l2SER']
      memSER = joinOptTable[app][vol][csize]['memSER']
      runtime = joinOptTable[app][vol][csize]['runtime']
      power = pwrCore + pwrL2 + pwrMem
      energy = power * runtime
      #Part 2: Filter RPPP by performance and reliability constraints
      if (runtime <= maxRuntime ):
        if (l1iSER <= maxErrRate and l1dSER <= maxErrRate):
          if (l2SER > maxErrRate):
            pwrL2Prot = pwrL2*l2EccCost
            l2Prot = 1
          else:
            pwrL2Prot = pwrL2
            l2Prot = 0
          if (memSER > maxErrRate):
            pwrMemProt = pwrMem*memEccCost
            memProt = 1
          else:
            pwrMemProt = pwrMem
            memProt = 0
          power = pwrCore + pwrL2Prot + pwrMemProt
          energy = power * runtime
          #Part 3: looking for a RPPP with minimum energy
          if energy < minEnergy:
          minEnergy = energy
          appConfVol = vol
          appConfCsize = csize
          appConfL2Prot = l2Prot
          appConfMemProt = memProt
```

References

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

[2] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold cmos technology," in *DAC '97: Proceedings of the 34th annual Design Automation Conference*. ACM, 1997, pp. 409–414.

[3] P. A. Chandrakasan, *Low Power Digital CMOS Design*. Kluwer Academic, 1995.

[4] S.-H. Lo, D. Buchanan, Y. Taur, and W. Wang, "Quantum-mechanical modeling of electron tunneling current from the inversion layer of ultra-thin-oxide nMOSFET's," *Electron Device Letters, IEEE*, vol. 18, no. 5, pp. 209–211, May. 1997.

[5] M. Green, E. Gusev, R. Degraeve, and E. Garfunkel, "Ultrathin ($< 4nm$) SiO_2 and Si-O-N gate dielectric layers for silicon microelectronics: Understanding the processing, structure, and physical and electrical limits," *Journal of Applied Physics*, vol. 90, no. 5, pp. 2057–2121, 2001.

[6] G. Ribes, J. Mitard, M. Denais, S. Bruyere, F. Monsieur, C. Parthasarathy, E. Vincent, and G. Ghibaudo, "Review on high-k dielectrics reliability issues," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 1, pp. 5–19, 2005.

[7] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305 – 316, Sep. 2005.

[8] D. Hocevar, P. Cox, and P. Yang, "Parametric yield optimization for mos circuit blocks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 7, no. 6, pp. 645 –658, jun 1988.

[9] X. Tang, V. De, and J. Meindl, "Intrinsic MOSFET parameter fluctuations due to random dopant placement," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 369 –376, dec. 1997.

[10] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Lifetime reliability: Toward an architectural solution," *IEEE Micro*, vol. 25, no. 3, pp. 70–80, 2005.

[11] 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.

[12] V. Chandra and R. Aitken, "Impact of voltage scaling on nanoscale sram reliability," in *DATE 2009.*, Apr. 2009, pp. 387–392.

[13] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.

[14] G. Tellez, A. Farrahi, and M. Sarrafzadeh, "Activity-driven clock design for low power circuits," in *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, 1995, pp. 62–65.

[15] L. Benini and G. De Micheli, "Transformation and synthesis of fsms for low-power gated-clock implementation," in *ISLPED '95: Proc. international symposium on Low power design.* New York, NY, USA: ACM, 1995, pp. 21–26.

[16] V. Tiwari, D. Singh, S. Rajgopal, R. Mehta, Gauravand Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *DAC '98: Proceedings of the 35th annual Design Automation Conference.* ACM, 1998, pp. 732–737.

[17] Q. Wu, P. Juang, M. Martonosi, and W. Clark, Douglas, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," *SIGOPS Oper. Syst. Rev.*, vol. 38, no. 5, pp. 248–259, 2004.

[18] H. Mahmoodi, V. Tirumalashetty, M. Cooke, and K. Roy, "Ultra low-power clocking scheme using energy recovery and clock gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 33–44, 2009.

[19] L. Benini, G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of synchronous controllers," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 4, no. 4, pp. 351–375, 1999.

[20] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on rtl clock-gating," in *DAC '03: Proceedings of the 40th annual Design Automation Conference.* ACM, 2003, pp. 622–627.

[21] J. J. Chen, X. Wei, Y. J. Jiang, and Q. Zhou, "Improve clock gating through power-optimal enable function selection," in *Design and Diagnostics of Electronic Circuits & Systems, 2009. DDECS '09. 12th International Symposium on*, Apr. 2009, pp. 30–33.

[22] Synopsys, *Power Compiler User Guide*, 2011.

[23] Y. Taur and T. Ning, *Fundamentals of Modern VLSI Devices.* Cambridge University Press, 2009.

[24] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 11, pp. 1571–1580, Nov. 2000.

[25] M. T. Schmitz and B. M. Al-Hashimi, "Considering power variations of dvs processing elements for energy minimisation in distributed systems," in *ISSS '01: Proceedings of the 14th international symposium on Systems synthesis*. New York, NY, USA: ACM, 2001, pp. 250–255.

[26] G. Semeraro, D. H. Albonesi, S. G. Dropsho, G. Magklis, S. Dwarkadas, and M. L. Scott, "Dynamic frequency and voltage control for a multiple clock domain microarchitecture," in *MICRO 35: Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 356–367.

[27] E. Talpes and D. Marculescu, "Toward a multiple clock/voltage island design style for power-aware processors," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 5, pp. 591–603, 2005.

[28] Y. Zhu, D. Albonesi, and A. Buyuktosunoglu, "A high performance, energy efficient gals processor microarchitecture with reduced implementation complexity," in *Performance Analysis of Systems and Software, IEEE International Symposium on*, Mar. 2005, pp. 42–53.

[29] G. Magklis, P. Chaparro, J. González, and A. González, "Independent front-end and back-end dynamic voltage scaling for a gals microarchitecture," in *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 2006, pp. 49–54.

[30] ESIA, JEITA, KSIA, TSIA, and SIA, "International technology roadmap for process integration, devices and structures," 2011.

[31] D. A. Muller, T. Sorsch, S. Moccio, F. H. Baumann, K. Evans-Lutterodt, and G. Timp, "The electronic structure at the atomic scale of ultrathin gate oxides," *Nature*, vol. 399, pp. 758–761, Jun. 1999.

[32] S. Max, "The end of the road for silicon," *Nature*, vol. 399, pp. 729–730, Jun. 1999.

[33] S. G. Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies*. Springer, 2005.

[34] J. Robertson, "High dielectric constant gate oxides for metal oxide si transistors," *Reports on Progress in Physics*, vol. 69, no. 2, pp. 327–396, 2006.

[35] L. Q. Wei, Z. P. Chen, M. Johnson, K. Roy, and V. De, "Design and optimization of low voltage high performance dual threshold cmos circuits," in *Design Automation Conference, 1998. Proceedings*, Jun. 1998, pp. 489–494.

[36] R. Chau, J. Brask, S. Datta, G. Dewey, M. Doczy, B. Doyle, J. Kavalieros, B. Jin, M. Metz, A. Majumdar, and M. Radosavljevic, "Application of high-k gate dielectrics and metal gate electrodes to enable silicon and non-silicon logic nanotechnology," *Microelectronics Engineering*, pp. 1–6, Jun. 2005.

[37] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-v high-speed mtcmos circuit scheme for power-down application circuits," *Solid-State circuits, IEEE Journal of*, vol. 32, no. 6, pp. 861–869, 1997.

[38] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual for System-on-Chip Design*. Springer, 2007.

[39] L. T. Clark, M. Kabir, and J. E. Knudsen, "A low standby power flip-flop with reduced circuit and control complexity," in *Proc. IEEE Custom Integrated Circuits Conference CICC '07*, 2007, pp. 571–574.

[40] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Drowsy instruction caches. leakage power reduction using dynamic voltage scaling and cache sub-bank prediction," in *Proc. 35th Annual IEEE/ACM International Symposium on (MICRO-35)Microarchitecture*, 2002, pp. 219–230.

[41] K. Kumagai, H. Iwaki, H. Yoshida, H. Suzuki, T. Yamada, and S. Kurosawa, "A novel powering-down scheme for low vt cmos circuits," in *VLSI Circuits, 1998. Digest of Technical Papers. 1998 Symposium on*, Jun. 1998, pp. 44–45.

[42] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power gating with multiple sleep modes," in *proc. Quality Electronic Design, 7th International Symposium on*, Mar. 2006, pp. 5pp.–637.

[43] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Addison-Wesley Pub. Co., 1993.

[44] M. Abriamovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. Wiley-Blackwell, 1994.

[45] R. Wadsack, "Fault modeling and logic simulation of cmos and mos integrated circuits," *AT T Technical Journal*, vol. 57, pp. 1449–1474, 1978.

[46] M. Bushnell and V. Agrawal, *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits*. Springer Netherlands, 2000.

[47] J. Emmert, C. Stroud, and J. Bailey, "A new bridging fault model for more accurate fault behavior," in *Proc. of the Design Automation Conf.*, 2000, pp. 481–485.

[48] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara, "Invisible delay quality-sdqm model lights up what could not be seen," in *Proc. of the International Test Conf.*, Nov. 2005, pp. 9 pp. – 1210.

[49] S. Zafar, "A model for negative bias temperature instability in oxide and high k pfets," in *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, May. 2007, pp. 1 –5.

[50] B. Kaczer, R. Degraeve, M. Rasras, K. Van de Mieroop, P. Roussel, and G. Groeseneken, "Impact of mosfet gate oxide breakdown on digital circuit operation and reliability," *Electron Devices, IEEE Transactions on*, vol. 49, no. 3, pp. 500 –506, Mar. 2002.

[51] J. Black, "Electromigration failure modes in aluminum metallization for semiconductor devices," *Proceedings of the IEEE*, vol. 57, no. 9, pp. 1587 – 1594, sept. 1969.

[52] K. N. Tu, "Recent advances on electromigration in very-large-scale-integration of interconnects," *Journal of Applied Physics*, vol. 94, no. 9, pp. 5451 –5473, Nov 2003.

[53] C.-Y. Li, P. Borgesen, and T. D. Sullivan, "Stress migration related electromigration damage mechanism in passivated, narrow interconnects," *Applied Physics Letters*, vol. 59, no. 12, pp. 1464 –1466, sep 1991.

[54] J. Pang, K. H. Tan, X. Shi, and Z. Wang, "Thermal cycling aging effects on microstructural and mechanical properties of a single pbga solder joint specimen," *Components and Packaging Technologies, IEEE Transactions on*, vol. 24, no. 1, pp. 10 –15, Mar. 2001.

[55] Y.-S. Lai, T. H. Wang, and C.-C. Lee, "Thermal mechanical coupling analysis for coupled power- and thermal-cycling reliability of board-level electronic packages," *Device and Materials Reliability, IEEE Transactions on*, vol. 8, no. 1, pp. 122 –128, march 2008.

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

[57] K. Gala, D. Blaauw, J. Wang, V. Zolotov, and M. Zhao, "Inductance 101: analysis and design issues," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 329 – 334.

[58] B. Kleveland, X. Qi, L. Madden, R. Dutton, and S. Wong, "Line inductance extraction and modeling in a real chip with power grid," in *Electron Devices Meeting, 1999. IEDM Technical Digest. International*, 1999, pp. 901 –904.

[59] N. Arora, L. Song, S. Shah, A. Sinha, and V. Chang, "Test chip for inductance characterisation and modeling for sub-100nm x architecture and manhattan chip

design," in *Microelectronic Test Structures, 2005. ICMTS 2005. Proceedings of the 2005 International Conference on*, april 2005, pp. 251 – 255.

[60] M. Elzinga, E. Chiprout, C. Dike, M. Wolfe, and M. Kobrinsky, "An active 90nm inductive signal noise testchip with realistic microprocessor signal buses," in *Integrated Circuit Design and Technology, 2006. ICICDT '06. 2006 IEEE International Conference on*, 0-0 2006, pp. 1 –5.

[61] M. Pant, P. Pant, D. Wills, and V. Tiwari, "Inductive noise reduction at the architectural level," in *VLSI Design, 2000. Thirteenth International Conference on*, 2000, pp. 162 –167.

[62] P. Sweeney, *Error Control Coding: From Theory to Practice.* Wiley, 2002.

[63] S. Lin and J. D. Costello, *Error Control Coding.* Prentice Hall, 2004.

[64] TSMC, *Cyclic Redundancy Check Computation: An Implementation Using the TMS320C54x*, 1999.

[65] S. Kim, S. V. Kosonocky, and D. R. Knebel, "Understanding and minimizing ground bounce during mode transition of power gating structures," in *Proc. International Symposium on Low Power Electronics and Design ISLPED '03*, 2003, pp. 22–25.

[66] S. Kim, S. V. Kosonocky, D. R. Knebel, and K. Stawiasz, "Experimental measurement of a novel power gating structure with intermediatepower saving mode," in *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design.* New York, NY, USA: ACM, 2004, pp. 20–25.

[67] S. Henzler, G. Georgakos, J. Berthold, and M. Eireiner, "Activation technique for sleep-transistor circuits for reduced powersupply noise," in *Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedingsof the 32nd European*, Sep. 2006, pp. 102–105.

[68] R. Bhanuprakash, M. Pattanaik, S. Rajput, and K. Mazumdar, "Analysis and reduction of ground bounce noise and leakage current during mode transition of stacking power gating logic circuits," in *TENCON 2009 - 2009 IEEE Region 10 Conference*, Jan. 2009, pp. 1 –6.

[69] M. H. Chowdhury, J. Gjanci, and P. Khaled, "Controlling ground bounce noise in power gating scheme for system-on-a-chip," in *Symposium on VLSI, 2008. ISVLSI'08. IEEE Computer Society Annual.* IEEE, 2008, pp. 437–440.

[70] H. Jiao and V. Kursun, "Ground bouncing noise suppression techniques for data preserving sequential mtcmos circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 763–773, 2011.

[71] Y. Sun, L. Xiao, and Y. Liu, "Ground bounce reduction in power gating circuits using input vector control," in *Laser Physics and Laser Technologies (RCSLPLT) and 2010 Academic Symposium on Optoelectronics Technology (ASOT), 2010 10th Russian-Chinese Symposium on*, 2010, pp. 345–348.

[72] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Enhancing the efficiency of energy-constrained dvfs designs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2012.

[73] P. Pillamari, K. Naidu, and H. Kittur, "Power reduction using dvfs with a producer-consumer fifo," in *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on*, 2011, pp. 454–458.

[74] A. Wang and A. Chandrakasan, "A 180-mv subthreshold fft processor using a minimum energy design methodology," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[75] B. Zhai, S. Hanson, D. Blaauw, and D. Sylvester, "Analysis and mitigation of variability in subthreshold design," in *Low Power Electronics and Design, 2005. ISLPED '05. Proceedings of the 2005 International Symposium on*, 2005, pp. 20–25.

[76] T. Enomoto, T. Oka, and H. Shikano, "A self-controllable voltage lavel (svl) circuit and its low-power high-speed cmos circuit applications," *JSSC*, vol. 38, no. 7, pp. 1220–1226, 2003.

[77] R. Krishnamurthy, A. Alvandpour, V. De, and S. Borkar, "High-performance and low-power chanllenges for sub-70nm microprocessor circuits," in *CICC 2002*, 2002, pp. 125–128.

[78] J. Wang and B. Calhoun, "Canary replica feedback for near-drv standby vdd scaling in a 90nm sram," in *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE*, 2007, pp. 29–32.

[79] J. Wang and B. H. Calhoun, "Techniques to extend canary-based standby scaling for srams to 45 nm and beyond," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 11, pp. 2514–2523, 2008.

[80] A. Sasan, K. Amiri, H. Homayoun, A. M. Eltawil, and F. J. Kurdahi, "Variation trained drowsy cache (vtd-cache): A history trained variaion aware drowsy cache for fine grain voltage scaling," *TVLSI*, no. 4, pp. 630–642, April 2012.

[81] A. Nourivand, A. J. Al-Khalili, and Y. Savaria, "Postsilicon tuning of standby supply voltage in srams to reduce yield losses due to parametric data-retention failure," *TVLSI*, no. 1, pp. 29–41, Jan 2012.

[82] A. Kumar, H. Qin, P. Ishwar, J. Rabaey, and K. Ramchandran, "Fundamental data retention limits in sram standby experimental results," in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, march 2008, pp. 92 –97.

[83] B. Calhoun and A. Chandrakasan, "Standby power reduction using dynamic voltage scaling and canary flip-flop structures," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 9, pp. 1504 – 1511, Sept. 2004.

[84] B. H. Calhoun and A. P. Chandrakasan, "A 256-kb 65-nm sub-threshold sram design for ultra-low-voltage operation," *Solid-State circuits, IEEE Journal of*, vol. 42, no. 3, pp. 680–688, 2007.

[85] T. M. Mak and S. Nassif, "Guest editors' introduction: Process variation and stochastic design and test," *Design Test of Computers, IEEE*, vol. 23, no. 6, pp. 436–437, 2006.

[86] L. R. Harriott, "Limits of lithography," *Proceedings of the IEEE*, vol. 89, no. 3, pp. 366–374, 2001.

[87] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 2, pp. 183 –190, Feb. 2002.

[88] 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.

[89] D. Reid, C. Millar, G. Roy, S. Roy, and A. Asenov, "Analysis of threshold voltage distribution due to random dopants: A 100 2009,000-sample 3-d simulation study," *Electron Devices, IEEE Transactions on*, vol. 56, no. 10, pp. 2255–2263, 2009.

[90] W. Zhao and et al., "Rigorous extraction of process variations for 65-nm CMOS design," *Semi. Manufac., IEEE Trans.*, vol. 22, no. 1, pp. 196–203, Feb. 2009.

[91] D. Reid, C. Millar, S. Roy, and A. Asenov, "Understanding ler-induced mosfet variability;part i: Three-dimensional simulation of large statistical samples," *Electron Devices, IEEE Transactions on*, vol. 57, no. 11, pp. 2801–2807, 2010.

[92] H. Wong, Y. Taur, and D. Frank, "Discrete random dopant distribution effects in nanometer-scale {MOSFETs}," *Microelectronics Reliability*, vol. 38, no. 9, pp. 1447 – 1456, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026271498000535

[93] K. Nishinohara, N. Shigyo, and T. Wada, "Effects of microscopic fluctuations in dopant distributions on mosfet threshold voltage," *Electron Devices, IEEE Transactions on*, vol. 39, no. 3, pp. 634–639, 1992.

[94] S. Kundu, A. Sreedhar, and A. Sanyal, "Forbidden pitches in sub-wavelength lithography and their implications on design," *Journal of computer-aided materials design*, vol. 14, no. 1, pp. 79–89, 2007.

[95] C. A. Mack, *Field guide to optical lithography*. SPIE Press Bellingham, Washington, USA, 2006.

[96] P. Oldiges, Q. Lin, K. Petrillo, M. Sanchez, M. Ieong, and M. Hargrove, "Modeling line edge roughness effects in sub 100 nanometer gate length devices," in *Simulation of Semiconductor Processes and Devices, 2000. SISPAD 2000. 2000 International Conference on*. IEEE, 2000, pp. 131–134.

[97] Y. Ye, T. Liu, M. Chen, S. Nassif, and Y. Cao, "Statistical modeling and simulation of threshold variation under random dopant fluctuations and line-edge roughness," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 6, pp. 987–996, 2011.

[98] A. Asenov, A. Brown, J. Davies, S. Kaya, and G. Slavcheva, "Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale mosfets," *Electron Devices, IEEE Transactions on*, vol. 50, no. 9, pp. 1837–1852, 2003.

[99] F. Wang and V. Agrawal, "Single event upset: An embedded tutorial," in *VLSI Design, 2008. VLSID 2008. 21st International Conference on*, 4-8 2008, pp. 429 –434.

[100] P. E. Dodd and F. W. Sexton, "Critical charge concepts for cmos srams," *Nuclear Science, IEEE Transactions on*, vol. 42, no. 6, pp. 1764–1771, 1995.

[101] 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.

[102] J. Ziegler and W. Lanford, "Effect of cosmic rays on computer memories," *Science*, vol. 206, no. 4420, p. 776, 1979.

[103] J.-M. Palau, G. Hubert, K. Coulie, B. Sagnes, M.-C. Calvet, and S. Fourtine, "Device simulation study of the seu sensitivity of srams to internal ion tracks generated by nuclear reactions," *Nuclear Science, IEEE Transactions on*, vol. 48, no. 2, pp. 225–231, 2001.

[104] P. Roche, J.-M. Palau, G. Bruguier, C. Tavernier, R. Ecoffet, and J. Gasiot, "Determination of key parameters for seu occurrence using 3-d full cell sram simulations," *Nuclear Science, IEEE Transactions on*, vol. 46, no. 6, pp. 1354–1362, 1999.

[105] Y. Tosaka, H. Kanata, S. Satoh, and T. Itakura, "Simple method for estimating neutron-induced soft error rates based on modified bgr model," *Electron Device Letters, IEEE*, vol. 20, no. 2, pp. 89 –91, Feb. 1999.

[106] K. Hass, R. Treece, and A. Giddings, "A radiation-hardened 16/32-bit microprocessor," *Nuclear Science, IEEE Transactions on*, vol. 36, no. 6, pp. 2252–2257, 1989.

[107] T. Karnik, S. Vangal, V. Veeramachaneni, P. Hazucha, V. Erraguntla, and S. Borkar, "Selective node engineering for chip-level soft error rate improvement [in cmos]," in *VLSI Circuits Digest of Technical Papers, 2002. Symposium on*, 2002, pp. 204–205.

[108] P. Hazucha, T. Karnik, S. Walstra, B. Bloechel, J. Tschanz, J. Maiz, K. Soumyanath, G. Dermer, S. Narendra, V. De, and S. Borkar, "Measurements and analysis of ser-tolerant latch in a 90-nm dual-vt cmos process," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 9, pp. 1536–1543, Sept. 2004.

[109] S. Shirinzadeh and R. Niarakiasli, "A novel soft error hardened latch design in 90nm cmos," in *Computer Architecture and Digital Systems (CADS), 2012 16th CSI International Symposium on*, 2012, pp. 60–63.

[110] V. Stojanovic, R. Iris Bahar, J. Dworak, and R. Weiss, "A cost-effective implementation of an ecc-protected instruction queue for out-of-order microprocessors," in *Proc. 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 705–708.

[111] A. Ejlali, B. Al-Hashimi, M. Schmitz, P. Rosinger, and S. 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, Apr. 2006.

[112] D. Alnajjar, M. Hashimoto, T. Onoye, and Y. Mitsuyama, "Static voltage overscaling and dynamic voltage variation tolerance with replica circuits and time redundancy in reconfigurable devices," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, 2012, pp. 1–7.

[113] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, "Fault injection into vhdl models: the mefisto tool," in *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on.* IEEE, 1994, pp. 66–75.

[114] M. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," *Transaction on Computers*, vol. 30, no. 4, pp. 75–82, 1997.

[115] J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs, and G. Leber, "Comparison of physical and software-implemented fault injection techniques," *Computers, IEEE Transactions on*, vol. 52, no. 9, pp. 1115 – 1133, sept. 2003.

[116] H. Zarandi, S. Miremadi, and A. Ejlali, "Fault injection into verilog models for dependability evaluation of digital systems," in *Proceedings of the Second international conference on Parallel and distributed computing*. IEEE Computer Society, 2003, pp. 281–287.

[117] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003, p. 29.

[118] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. Mukherjee, and R. Rangan, "Computing architectural vulnerability factors for address-based structures," in *Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on*, 4-8 2005, pp. 532 – 543.

[119] A. Biswas, P. Racunas, J. Emer, and S. Mukherjee, "Computing accurate avfs using ace analysis on performance models: A rebuttal," *Computer Architecture Letters*, vol. 7, no. 1, pp. 21 –24, january-june 2008.

[120] 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.

[121] W. Zhang, "Computing cache vulnerability to transient errors and its implication," in *Defect and Fault Tolerance in VLSI Systems, 20th IEEE International Symposium on*, Oct. 2005, pp. 427 – 435.

[122] G.-H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli, "Balancing performance and reliability in the memory hierarchy," in *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 269–279.

[123] A. Biswas, C. Recchia, S. Mukherjee, V. Ambrose, L. Chan, A. Jaleel, A. Papathanasiou, M. Plaster, and N. Seifert, "Explaining cache ser anomaly using due avf measurement," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, 9-14 2010, pp. 1 –12.

[124] S. Wang, J. Hu, and S. Ziavras, "On the characterisation and optimization of on-chip cache reliability against soft errors," *Computers, IEEE Transactions on*, vol. 58, no. 9, pp. 1171 –1184, Sep. 2009.

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

[126] J. Hu, S. Wang, and S. Ziavras, "On the exploitation of narrow-width values for improving register file reliability," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 7, pp. 953–963, 2009.

[127] M. Kandala, W. Zhang, and L. Yang, "An area-efficient approach to improving register file reliability against transient errors," in *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, vol. 1, 2007, pp. 798–803.

[128] 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*, 1999, pp. 84–91.

[129] S. Reinhardt and S. Mukherjee, "Transient fault detection via simultaneous multithreading," in *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, 2000, pp. 25–36.

[130] S. Mukherjee, M. Kontz, and S. Reinhardt, "Detailed design and evaluation of redundant multi-threading alternatives," in *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, 2002, pp. 99–110.

[131] T. Vijaykumar, I. Pomeranz, and K. Cheng, "Transient-fault recovery using simultaneous multithreading," in *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, 2002, pp. 87–98.

[132] M. Gomaa, C. Scarbrough, T. Vijaykumar, and I. Pomeranz, "Transient-fault recovery for chip multiprocessors," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, 2003, pp. 98–109.

[133] A. Mendelson and N. Suri, "Designing high-performance and reliable superscalar architectures-the out of order reliable superscalar (o3rs) approach," in *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*, 2000, pp. 473–481.

[134] J. Ray, J. Hoe, and B. Falsafi, "Dual use of superscalar datapath for transient-fault detection and recovery," in *Microarchitecture, 2001. MICRO-34. Proceedings. 34th ACM/IEEE International Symposium on*, 2001, pp. 214–224.

[135] A. Parashar, S. Gurumurthi, and A. Sivasubramaniam, "A complexity-effective approach to alu bandwidth enhancement for instruction-level temporal redundancy," in *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, 2004, pp. 376–386.

[136] J. Smolens, J. Kim, J. Hoe, and B. Falsafi, "Efficient resource sharing in concurrent error detecting superscalar microarchitectures," in *Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on*, 2004, pp. 257–268.

[137] M. Gomaa and T. N. Vijaykumar, "Opportunistic transient-fault detection," *Micro, IEEE*, vol. 26, no. 1, pp. 92–99, 2006.

[138] 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.

[139] L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "Soft error and energy consumption interactions: a data cache perspective," in *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, 2004, pp. 132–137.

[140] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, 2007, pp. 197–209.

[141] W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam, "Icr: in-cache replication for enhancing data cache reliability," in *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, 2003, pp. 291–300.

[142] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002, pp. 1567–1576.

[143] X. Ning and C. G. Cassandras, "Optimal dynamic sleep time control in wireless sensor networks," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2008, pp. 2332–2337.

[144] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-v high-speed mtcmos circuit scheme for power-down application circuits," *Solid-State Circuits, IEEE Journal of*, vol. 32, no. 6, pp. 861–869, 1997.

[145] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle mac with scheduled channel polling," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006, pp. 321–334.

[146] C. Zeng, N. Saxena, and E. J. McCluskey, "Finite state machine synthesis with concurrent error detection," in *Test Conference, 1999. Proceedings. International*, 1999, pp. 672–679.

[147] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?" in *Test Conference, 2000. Proceedings. International*. IEEE, 2000, pp. 985–994.

[148] K. Furutani, K. Arimoto, H. Miyamoto, K. Kobayashi, T. andYasuda, and K. Mashiko, "A built-in hamming code ecc circuit for drams," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 50–56, 1989.

[149] S. Gerstendorfer and H.-J. Wunderlich, "Minimised power consumption for scan-based bist," in *Test Conference, 1999. Proceedings. International*, 1999, pp. 77 –84.

[150] ARM, "Arm cortex-m0 description." [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-m0.php

[151] N. Seifert, V. Ambrose, B. Gill, Q. Shi, R. Allmon, C. Recchia, S. Mukherjee, N. Nassif, J. Krause, J. Pickholtz, and A. Balasubramanian, "On the radiation-induced soft error performance of hardened sequential elements in advanced bulk cmos technologies," in *International Reliability Physics Symposium*, 2010.

[152] Tezzaron, "Soft errors in electronic memory a white paper," Tezzaron, Tech. Rep., 2004. [Online]. Available: http://www.tezzaron.com/about/papers/soft_errors_1_1_secure.pdf

[153] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, and J. Gill, B.and Maiz, "Radiation-induced soft error rates of advanced cmos bulk devices," in *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, Mar. 2006, pp. 217 –225.

[154] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[155] L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "Soft error and energy consumption interactions: a data cache perspective," in *Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on*, 2004, pp. 132–137.

[156] M. Tahoori, "Defects, yield, and design in sublithographic nano-electronics," in *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, 3-5 2005, pp. 3 – 11.

[157] S. Andrew, *Computer Networks*. Prentics Hall, 1996.

[158] USB-IF, *Universal Serial Bus Specification*. [Online]. Available: http://www.usb.org/developers/docs/

[159] G. Albertengo and R. Sisto, "Parallel crc generation," *Micro, IEEE*, vol. 10, no. 5, pp. 63 –71, Oct. 1990.

[160] J. Autran, P. Roche, S. Sauze, G. Gasiot, D. Munteanu, P. Loaiza, M. Zampaolo, and J. Borel, "Altitude and underground real-time ser characterisation of cmos 65nmsram," *Nuclear Science, IEEE Transactions on*, vol. 56, no. 4, pp. 2258 – 2266, Aug. 2009.

[161] A. Saleh, J. Serrano, and J. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *Reliability, IEEE Transactions on*, vol. 39, no. 1, pp. 114 –122, Apr. 1990.

[162] V. Chandra and R. Aitken, "Impact of technology and voltage scaling on the soft error susceptibility in nanoscale cmos," in *Defect and Fault Tolerance of VLSI Systems, 2008. DFTVS '08. IEEE International Symposium on*, Oct. 2008, pp. 114–122.

[163] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *Dependable Systems and Networks, 2004 International Conference on*, 2-8 2004, pp. 177 – 186.

[164] S. Sze and K. Ng, *Physics of Semiconductor Devices*. John Wiley & Sons, 2007.

[165] UPF, "Ieee standard for design and verification of low power integrated circuits," *IEEE Std 1801-2013*, pp. 1–218, 2013.

[166] O. Oklahoma State Univ., Stillwater, "Osu freepdk," 2013, *http://vlsiarch.ecen.okstate.edu/flows/*.

[167] S. Sun and P. Tsui, "Limitation of cmos supply-voltage scaling by mosfet threshold-voltage variation," *Solid-State Circuits, IEEE Journal of*, vol. 30, no. 8, pp. 947–949, 1995.

[168] I. Koren and M. Krishna, *Fault-Tolerant Systems*. Morgan-Kaufman, 2007.

[169] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, 2007, pp. 197–209.

[170] C. Chen and M. 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, 1984.

[171] D. Keitel-Schulz and N. Wehn, "Embedded dram development: Technology, physical design, and application issues," *Design Test of Computers, IEEE*, vol. 18, no. 3, pp. 7–15, 2001.

[172] K. Nii, Y. Tsukamoto, T. Yoshizawa, S. Imaoka, Y. Yamagami, T. Suzuki, A. Shibayama, H. Makino, and S. Iwade, "A 90-nm low-power 32-kb embedded sram with gate leakage suppression circuit for mobile applications," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 4, pp. 684–693, 2004.

[173] Y. Ishii, H. Fujiwara, K. Nii, H. Chigasaki, O. Kuromiya, T. Saiki, A. Miyanishi, and Y. Kihara, "A 28-nm dual-port sram macro with active bitline equalizing circuitry against write disturb issue," in *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, 2010, pp. 99–100.

[174] R. Kalla, B. Sinharoy, W. Starke, and M. Floyd, "Power7: Ibm's next-generation server processor," *Micro, IEEE*, vol. 30, no. 2, pp. 7–15, 2010.

[175] T. Instrument, "Omap4460 multimedia device technical reference manual," 2013, *http://www.ti.com/product/omap4430*.

[176] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. Hill, and D. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[177] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterisation, 2001. WWC-4. 2001 IEEE International Workshop on*. IEEE, 2001, pp. 3–14.

[178] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, dec. 2009, pp. 469 –480.

[179] Linaro, "Linaro gcc toolchain," *https://wiki.linaro.org/WorkingGroups/ToolChain*.

[180] P. Hazucha, T. Karnik, B. Bloechel, C. Parsons, D. Finan, and S. Borkar, "Area-efficient linear regulator with ultra-fast load regulation," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 4, pp. 933–940, 2005.

[181] R. Milliken, J. Silva-Martinez, and E. Sanchez-Sinencio, "Full on-chip cmos low-dropout voltage regulator," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 9, pp. 1879–1890, 2007.

[182] L. Chang, R. K. Montoye, Y. Nakamura, K. A. Batson, R. J. Eickemeyer, R. H. Dennard, W. Haensch, and D. Jamsek, "An 8t-sram for variability tolerance and low-voltage operation in high-performance caches," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 4, pp. 956–963, 2008.

[183] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu, "Trading off cache capacity for reliability to enable low voltage operation," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*. IEEE, 2008, pp. 203–214.

[184] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, april 2011, pp. 5B.4.1 –5B.4.7.

[185] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 4, pp. 473 –484, Apr. 1992.

[186] AMD, "Acp the truth about power consumption starts here," http://www.amd.com/uk/Documents/43761C_ACP_WP_EE.pdf, 2009.

[187] ARM, *ARM Cortex-M0 Technical Reference Manual*, ARM, 2009. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0432c/DDI0432C_cortex_m0_r0p0_trm.pdf

[188] J. Yiu, *The definitive guide to the ARM Cortex-M3*. Access Online via Elsevier, 2009.