AN ABSTRACT OF THE DISSERTATION OF

Joseph A. Crop for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on May 28, 2014.

Title: Methods to Improve the Reliability and Resiliency of Near/Sub-Threshold Digital Circuits

Abstract approved: _____

Patrick Yin Chiang

Energy consumption is one of the primary bottlenecks to both large and small scale modern compute platforms. Reducing the operating voltage of digital circuits to voltages where the supply voltage is near or below the threshold of the transistors has recently gained attention as a method to reduce the energy required for computations by as much as 6 times. However, when operating at near/sub-threshold voltages (where the supply voltage is near or below the threshold of the transistors), imperfections in transistor manufacturing, changes in temperature, and other difficult-to-predict factors cause wide variations in the timing of Complementary Metal-Oxide Semiconductor (CMOS) circuits due to an increased sensitivity at lower voltages. These increased variations result in poor aggregate performance and cause increased rates of error occurrence in computation.

This work introduces several new methods to improve the reliability of near/sub-threshold circuits. The first is a design automation technique that is used to aid in low-voltage digital standard cell synthesis. Second, two circuit-level techniques are also introduced that aim to improve the reliability and resiliency of digital circuits by means of completion/error detection. These techniques are shown to improve speed and lower energy consumption at low overheads compared to previous methods. Most importantly, these circuit-level methods are specifically designed to operate at low voltages and can themselves tolerate variations and operation in harsh environments. Finally, a test-chip prototype designed in 65nm-CMOS demonstrates the practicality and feasibility of a proposed current sensing error detector.

Methods to Improve the Reliability and Resiliency of Near/Sub-Threshold Digital Circuits

by

Joseph A. Crop

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 28, 2014
Commencement June 2014

Doctor of Philosophy dissertation of Joseph A. Crop presented on May 28, 2014.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Joseph A. Crop, Author

ACKNOWLEDGEMENTS

First of all, I would like to thank my adviser Patrick Chiang for providing me with the resources and guidance I needed to conduct this research. His non-traditional views on academia and the VLSI field have given me a valuable perspective I would not have received otherwise.

I would like to thank all of my collaborators around the world. Including Mattan Erez and Evgeni Krimer from The University of Texas at Austin. Their team provided valuable architecture modeling an analysis for the later portion of this research. My collaborators at Fudan University in Shanghai China who graciously hosted me for a summer: Junyu Wang, Jerry Fu, MengQin Xiao, and Hao Min. And finally my colleagues at Oregon State University. Namely Robert Pawlowski, Jacob Postman, Nariman Moezzi-Madani, Jarrod Jackson, and Scott Fairbanks. Without their supporting contributions to my research this dissertations would not be a success.

I would like to give a special thanks to my family for their support throughout my 29-year academic career which has been invaluable. Among them, Amy Kao and Lauri Shainsky helped edit this dissertation.

This work was funded in part by a grant from the Department of Energy Early Career Program, the Center for the Design of Analog and Digital Integrated Circuits (CDADIC), and gifts from Intel Corporation.

TABLE OF CONTENTS

Page

1 Introduction 1

2 Circuit Reliability Challenges in Near-Threshold 3
- 2.1 Introduction . . . 3
- 2.2 Near/Sub-Threshold Operation . . . 3
- 2.3 Variations at Near/Sub-Threshold . . . 5
- 2.4 Sources of Variation in CMOS circuits . . . 7
  - 2.4.1 Static Variations . . . 7
  - 2.4.2 Dynamic Variations . . . 8
- 2.5 Characteristics of Variations . . . 8
  - 2.5.1 Random Variations . . . 8
  - 2.5.2 Systematic Variations . . . 9
- 2.6 Soft Errors . . . 9
  - 2.6.1 Temperature-Induced Soft Errors . . . 9
  - 2.6.2 Voltage-Induced Soft Errors . . . 10
  - 2.6.3 Process Variation-Induced Soft Errors . . . 11
  - 2.6.4 Aging Effects on Soft Errors . . . 12
  - 2.6.5 Radiation-Induced Soft Errors . . . 12
- 2.7 Hard Errors . . . 13
- 2.8 Summary . . . 13

3 The Current Art in Circuit Reliability 15
- 3.1 Introduction . . . 15
- 3.2 Speculative Speed-Up and Error Detection Techniques . . . 16
  - 3.2.1 Architectural Retiming . . . 17
  - 3.2.2 Circuit-Level Speculation . . . 20
  - 3.2.3 Tunable Replica Circuits . . . 21
  - 3.2.4 Razor Flip-Flops . . . 22
  - 3.2.5 Transition Detectors . . . 24
- 3.3 Error Recovery Techniques . . . 26
  - 3.3.1 Clock Gating . . . 26
  - 3.3.2 Counterflow Pipelining . . . 28
  - 3.3.3 Micro-Rollback . . . 29
  - 3.3.4 Multiple Issue . . . 30

TABLE OF CONTENTS (Continued)

Page

3.3.5 Adaptive Scaling Methods . . . 33
3.3.6 Checkpoint-Restart . . . 34
3.3.7 Summary of Recovery Methods . . . 34
3.4 Higher Level Error-Protection Techniques . . . 35
3.4.1 Architecture Level Error-Protection Techniques . . . 36
3.4.2 Architectural Data Protection . . . 39
3.4.3 Software Level Error-Protection Techniques . . . 40
3.5 Summary Error Detection/Protection of Methods . . . 42
3.6 Error Detection and Recovery in Near/Sub-Threshold . . . 44
3.6.1 Speculative Error Detection in Near/Sub-Threshold . . . 45
3.6.2 Error Recovery Techniques in Near/Sub-Threshold . . . 46
3.6.3 Near/Sub-Threshold Timing Error Detection . . . 47
3.7 Conclusions . . . 47

4 A Design Automation Methodology Approach 49
4.1 Introduction . . . 49
4.2 Limitations of Existing Libraries and Design Techniques in Near/Sub-Threshold . . . 51
4.2.1 Combinational Logic Failure . . . 52
4.2.2 Sequential Logic Failure . . . 53
4.2.3 Timing Model Inaccuracy . . . 54
4.2.4 Design Time and Portability . . . 55
4.3 Proposed Near/Sub-Threshold Characterization Method . . . 56
4.3.1 Parse Standard Cell Liberty File . . . 56
4.3.2 Input-to-Output Delay Variation Test . . . 57
4.3.3 Analysis and Cell Removal Decisions . . . 58
4.4 Results . . . 61
4.4.1 Characteristics of Removed Cells . . . 62
4.4.2 Area Improvement . . . 62
4.4.3 Delay Improvement . . . 63
4.4.4 Energy Improvement . . . 65
4.5 Conclusions . . . 67

TABLE OF CONTENTS (Continued)

Page

5 Energy Constrained Encryption: A Case Study in Near-Threshold Circuit Design 68

5.1 Introduction . . . 68

5.2 Potential Solutions to Near-Threshold Challenges . . . 69
5.2.1 Check and Adapt . . . 70
5.2.2 Large Guard-Banding . . . 71
5.2.3 Architectural or Circuit Redesign . . . 71
5.2.4 Near-Threshold-Aware Synthesis Techniques . . . 72

5.3 Case-Study . . . 73
5.3.1 Hummingbird Encryption Scheme . . . 73
5.3.2 Tested Solutions . . . 74

5.4 Results . . . 79
5.4.1 Area Impact . . . 79
5.4.2 Energy and Power Impact . . . 80
5.4.3 Variation of Each Architecture . . . 81

5.5 Conclusion . . . 85

6 Asynchronous Circuit Operation 86

6.1 Introduction . . . 86

6.2 Traditional Asynchronous Techniques and Challenges . . . 87
6.2.1 Asynchronous Logic Elements . . . 87
6.2.2 Hazards . . . 88

6.3 Asynchronous Micropipelines . . . 89
6.3.1 Micropipeline Configurations . . . 89
6.3.2 Handshake Mechanisms . . . 90

6.4 Asynchronous Completion Detection . . . 91
6.4.1 Custom Circuits . . . 92
6.4.2 Null Convention Logic . . . 93
6.4.3 Matched Delay Lines . . . 93
6.4.4 Speculative Completion . . . 94
6.4.5 Current Sensing Completion Detection . . . 96
6.4.6 Sense-Inverter Based CSCD . . . 97
6.4.7 Activity-Monitoring Completion Detection (AMCD) . . . 99

6.5 Proposed Completion Detection Methods . . . 100

TABLE OF CONTENTS (Continued)

Page

6.5.1 Proposed Transition Aware Completion Detection 100
6.5.2 Proposed Pseudo-Asynchronous CSCD Method 102
6.6 Compared Results 103
6.7 Conclusions 106

7 Synchronous Completion Detection 107
7.1 Introduction 107
7.2 Variation Study 108
7.3 Proposed Error Detection Methods 110
7.3.1 Transition Detecting 110
7.3.2 Current Sensing Completion Detection 111
7.4 Results 116
7.4.1 Razor Results 117
7.4.2 TACD Results 117
7.4.3 CSCD Results 118
7.4.4 Energy, Area and Complexity 119
7.5 Conclusion 121

8 Current Sensing Completion Detection Test Chip 122
8.1 Introduction 122
8.2 Previous Work Comparison 122
8.3 CSCD Test Chip Design 125
8.3.1 SIMD Pipeline Design 126
8.3.2 Current Sensor Design 129
8.3.3 Design for Test Features 134
8.4 Chip Measurement Results 136
8.4.1 Sensor Calibration Analysis 137
8.4.2 Droop Plot Measurements 138
8.4.3 Noise Analysis 140
8.4.4 Power Gate Sizing 141
8.4.5 Throughput Analysis 142
8.4.6 Results Summary 144
8.5 Future Work 145

TABLE OF CONTENTS (Continued)

Page

8.5.1 Sensor Improvements . . . 145
8.5.2 System Improvements . . . 146
8.6 Conclusions . . . 146

9 Using Current Sensors to Detect Errors in a Radiation Environment 148
9.1 Introduction . . . 148
9.2 Related Work . . . 149
9.2.1 Current Sensors . . . 153
9.3 Current-Sensor SET Error Detector . . . 156
9.3.1 SET Detector Operating Principles . . . 156
9.3.2 Masking and False Alarms . . . 158
9.3.3 Sensor Calibration . . . 159
9.4 Architecture . . . 160
9.4.1 SET Detection and Recovery . . . 161
9.4.2 Interaction with Other Protection Mechanisms . . . 163
9.4.3 Calibration Architecture . . . 164
9.5 Methodology . . . 164
9.5.1 Current Sensor Evaluation Methodology . . . 164
9.5.2 False Alarm Evaluation Methodology . . . 165
9.6 Results . . . 167
9.6.1 Single Gate Analysis . . . 167
9.6.2 4-bit Ripple-Carry Adder . . . 168
9.6.3 False Alarm Rate Analysis and Performance Impact . . . 172
9.6.4 Implementation Overheads . . . 173
9.7 Conclusions . . . 176

10 Conclusions 178
10.1 Final Thoughts . . . 179
10.2 Future Work . . . 180

Bibliography 180

LIST OF FIGURES

Figure		Page
2.1	Impact of supply scaling on static timing analysis worst-case input on 16-bit multiplier in 45nm SOI. (a) delay; (b) energy/operation .	4
2.2	Delay variability as supply voltage is scaled to near-threshold for the adder in Fig. 2.1.	6
2.3	Delay variation widens as circuits are driven deeper into the sub-threshold region (Monte-Carlo simulation results from a Multiply-Accumulate in $45nm$ CMOS) [3].	7
3.1	Architectural Retiming applied to an subtraction unit, the use of a negative register and regular register simplify to a wire and do not slow the datapath because the negative register stores a decision made in the previous pipeline stage.	19
3.2	An example of a typical circuit-level speculation scheme: a speculator is used to speedup the system based on path activation probabilities from a model.	20
3.3	A Typical Tunable Replica Circuit (TRC).	21
3.4	(a) Synthesizable Razor error detector (b) Example timing diagram of error detection.	22
3.5	Razor II latch with detection clock generator and transition detector.	24
3.6	Two versions of modified Razor flip-flops: (a) Transition Detection with Time Borrowing (TDTB). (b) Double Sampling with Time Borrowing (DSTB).	25
3.7	(a) Pipeline modification for Clock Gating error recovery method. (b) Clock Gating pipeline data path with errors.	27
3.8	(a) Pipeline modification for Counterflow Pipelining error recovery method. (b) Counterflow pipeline data path with errors.	28
3.9	(a) Pipeline modification for Micro-Rollback error recovery method. (b) Micro-Rollback pipeline data path with errors.	31
3.10	(a) Pipeline modification for Multiple Issue error recovery method. (b) Multiple Issue pipeline data path with errors.	32

LIST OF FIGURES (Continued)

Figure Page

3.11 A systems-level diagram illustrating the feedback process of dynamically adapting circuit performance in the presence of dynamic variations. 33

3.12 Architecture Level Error-Protection Techniques. (a) Dual Modular Redundancy (DMR). (b) Triple Modular Redundancy (TMR). (c) Simplified Redundancy. 37

3.13 Delay variation widens as circuits are driven deeper into the sub-threshold region (Monte-Carlo simulation results from a Multiply-Accumulate in $45nm$ CMOS) [3]. 44

4.1 Flip-flop (a) contains ratio feedback and does not function properly at low sub-threshold voltages whereas flip-flop (b) contains tri-state feedback and does function properly at low sub-threshold voltages. . 54

4.2 Functional diagram of the method used to determine poor cells and their removal from the sub-threshold standard cell library. 57

4.3 Histogram of input-to-output delay variation from near/sub-threshold to nominal voltage in a low-power 90nm cell library. 59

4.4 Bar plot of input-to-output delay simulations from near/sub-threshold to nominal voltage in a low-power 90nm cell library. 60

4.5 Histograms of FP-ADD delay with and without cells removed from synthesizer using 1σ culling method running at Vdd=400mV. . . . 64

4.6 Box-plots of delay without and with cells removed from the synthesizer for both designs. 65

4.7 Box-plots of energy/computation without and with cells removed from the synthesizer. 66

5.1 A systems-level diagram illustrating the feedback process of dynamically adapting circuit performance in the presence of dynamic variations. 70

5.2 The Hummingbird cryptographic algorithm. 75

LIST OF FIGURES (Continued)

Figure		Page
5.3	Original implementation of the Hummingbird algorithm.	76
5.4	Area-optimized implementation of the Hummingbird algorithm.	77
5.5	Loop-unrolled implementation of the Hummingbird algorithm.	78
5.6	Power plot vs. time of test 0 (Shared EK) and test 2 (Speed Optimized).	81
5.7	Variation Response of four test cases across Monte Carlo.	82
6.1	Asynchronous Muller-C Element: (a) schematic symbol, (b) Synthesize-able, hazard-free circuit schematic [89] (c) C-element truth table.	88
6.2	A fundamental μ-pipe with 2-phase bundled data control via C-elements	89
6.3	(a) 4-phase and 2-phase bundled data handshaking conventions, (b) 4-phase to 2-phase conversion with the Toggle element, (c) 2-phase to 4-phase conversion with the merge (XOR) element.	91
6.4	An asynchronous μ-pipe with completion detection by means of a *matched delay line*.	94
6.5	An asynchronous μ-pipe with completion detection by means of *speculative completion detection*.	95
6.6	A standard combinational block modified for asynchronous operation by means of *current sensing completion detection*.	97
6.7	Basic configuration of *sense-inverter based current sensing completion detection* within an adder structure.	98
6.8	Basic configuration of *activity monitoring completion detection* within an adder structure.	99
6.9	(a) TACD within a μ-pipe stage, (b) Synthesize-able TACD schematic, (c) Timing diagram of TACD.	101
6.10	Synchronous CSCD method modified for asynchronous operation.	103

LIST OF FIGURES (Continued)

Figure | Page

6.11 Simulation of TACD, MDL, and CSCD with average-case operands into a 16-bit carry-save multiplier. 105

7.1 Histograms of (a) Monte Carlo chip-to-chip delay of the STA and (b) delay of changing FIR filter data on a 16-bit adder with error-detection speeds marked. 108

7.2 Potential throughput improvement with ideal error detection 109

7.3 (a) CSCD schematic, (b) CSCD timing diagram. 112

7.4 (a) Offset calibration of CSCD across 100 Monte Carlo simulations, (b) Speed of sensor stays relatively fast in NTV regime. 114

7.5 Power gate sizing has very minimal affect on speed while maintaining a reasonable sensing margin. 115

7.6 Input referred supply noise before and after calibrating CSCD sensor's RC noise filter. 116

7.7 Simulated throughput of TACD. 118

7.8 Simulated throughput of CSCD 119

8.1 Conceptual demonstration of: (a) Razor technique; (b) TRC technique; (c) Proposed current sensing completion detection technique. 124

8.2 Block diagram of test chip with sensors connected to SIMD lanes. . 125

8.3 Detailed architecture of pipeline stages: (a) Instruction Generator; (b) Register File; (c) Pipelined Array Multiplier (stage 1). 128

8.4 Schematic of current sensor used in test chip. 130

8.5 Flow chart of calibration procedure. 133

8.6 Die micro-photograph of test chip. 135

8.7 Lab test setup with custom designed test board and automated software running in LabView CVI. 137

8.8 Measured comparator offset before and after calibration. 138

LIST OF FIGURES (Continued)

Figure Page

8.9 On-die voltage droop plots captured using the analog sensors as on-die oscilloscopes, for logic operating at 1.0V and 0.75V. Arrows at top denote when logic operation has finished. 139

8.10 Simulated supply noise rejection at inputs of comparator after noise filtering. 140

8.11 Sensing margin impact on noise-induced false positives at 1V supply. 140

8.12 Measured on-die scatter plots of sensing droops before and after basic calibration of noise filtering. 141

8.13 Power gate sizing for required voltage droop vs. impact on speed reduction. 142

8.14 Comparison of throughput vs. energy efficiency at various supply voltages vs. conventional. 143

8.15 Throughput improvement beyond the EDA margin vs. conventional. 143

9.1 Block diagram of current sensor for SET detection. 154

9.2 SET strike at a closed NMOS transistor. 155

9.3 SET strike at closed PMOS transistor. 156

9.4 SET modeling in SPICE using a current source. 165

9.5 Minimum current pulse for an SET in an FO1 inverter (red); and a detectable pulse that is too small to generate an SET (green). . . . 166

9.6 Single gate response to an SET-generating strike current pulse. . . . 168

9.7 Single gate response to a low-energy particle strike current pulse. Top sub-figure is gate output and bottom sub-figure is the current sensor output. 169

9.8 Five simulated SET scenarios for a 4-bit adder. Critical path is highlighted, and the 5 strike scenarios are marked a–e (strike e is to an inverter within the XOR). 170

LIST OF FIGURES (Continued)

Figure Page

9.9 Adder response to five scenarios of SET-generating strike current pulses. 171

9.10 Three cycles of adder operation showing Cout and V_{sense}. Latching windows in grey, strike current duration within dotted lines. Current is sensed only within the grey latching window and normal currents are effectively masked. 172

9.11 True and false positive FIT rates (lower and upper part of bars on left axis) and false-alarm rate (dotted line on right axis) for ISCAS'89 benchmark circuits. The benchmark names are along the x-axis. 174

LIST OF TABLES

Table		Page
2.1	Classification of variations in digital logic.	14
3.1	Ability of different detection methods to work in various variation modes.	18
3.2	Comparison of Pipeline Recovery Methods.	35
3.3	Various ECC approaches overheads (based on [52])	43
3.4	Comparison of existing errors detection/prediction methods	43
4.1	Area Comparison of Culled Designs with Unculled Designs	63
4.2	Leakage energy comparison of culled designs with unculled designs.	66
5.1	Simulation results for each tested design	84
6.1	Comparison of different completion detection methods.	104
7.1	Comparison of error-detection methods.	120
8.1	Comparison of existing error-detection method with measured test chip.	123
8.2	Design summary of test chip.	144
9.1	Comparison of particle-strike induced error detection mechanisms for combinational logic.	151

to my family

Chapter 1: Introduction

Computing devices have reached a speed limitation due to a thermal limit manifested by having too many transistors in a chip. However, new applications such as wireless sensors and wearable devices have unlocked a new computing paradigm that doesn't rely on high speed, high power processing systems. These computing systems of the future are now energy-constrained by battery or energy harvesting requirements, usually being small amounts of energy with long lifespans. One such approach discussed in detail in this dissertation is the method of lowering the supply voltage of chips into the near-threshold region. In the past decade a number of researchers have sought to determine viable methods to make near-threshold computing reliable and achievable. However, a major concern with this approach is reliability. Ensuring reliable operation at low cost and overhead is critical in order to achieve near-threshold ubiquity in computing.

The primary goals of this dissertation are to explore, understand, and overcome the challenges of near-threshold digital logic design. In particular, this dissertation will examine five main research questions: (1) What are the main concerns with near-threshold circuit reliability? (2) How do environmental factors such as temperature and radiation manifest as errors in near-threshold systems? (3) Can

near-threshold operation of digital circuits be improved with synthesis-level techniques? (4) Can asynchronous methods be employed to provide reliable operation in near-threshold? (5) Can a system be designed that is reliable and resilient to every type of error source in near-threshold? This study makes a major contribution to the research of near-threshold computing by demonstrating several approaches that improve the reliability and resiliency of such circuits and systems.

The overall structure of this dissertation takes the form of ten chapters, including this introductory chapter. Chapter Two begins by investigating the current reliability challenges in modern CMOS and how they pertain to near-threshold operation. The third chapter introduces the the current art in digital circuit reliability and discusses current methods aimed at reliability. The fourth and fifth chapters present a proposed design automation methodology and a case study of its efficacy. Chapters 6, 7, and 8 introduce asynchronous circuits and how they can be adapted for use in synchronous digital systems, focusing on current sensing completion detection. Next an application of these adapted circuits are proposed for reliability in a radiation environment. Chapter 10 concludes with a brief summary and critique of the findings and includes a discussion of the implication of the findings to future research into this area while areas for further research are identified.

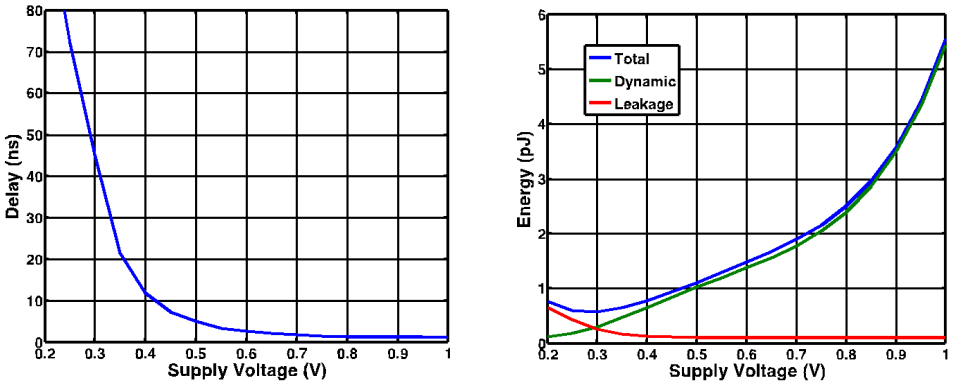
Chapter 2: Circuit Reliability Challenges in Near-Threshold

2.1 Introduction

This chapter's purpose is to introduce the concept of near-threshold operation and the circuit reliability challenges associated with it. It begins by discussing the attraction of near-threshold, focusing on the trade-offs of energy and delay. Next, a detailed discussion of the sources of variations in modern CMOS are presented. Each source is classified as it pertains to circuit reliability and the types of errors they might generate are discussed.

2.2 Near/Sub-Threshold Operation

One of the most popular methods to reduce power consumption is to aggressively lower the supply voltage into the sub-threshold or near-threshold voltage region. With near-threshold operation, the supply voltage is lowered to just above the threshold voltage of the transistors. This has been previously shown to lower energy by ~5-10X while decreasing the operating frequency by as much as ~10X.



(a) (b)

Figure 2.1: Impact of supply scaling on static timing analysis worst-case input on 16-bit multiplier in 45nm SOI. (a) delay; (b) energy/operation

In [1] it is shown that this region of operation provides the best energy savings without introducing significantly long delays, when compared to sub-threshold operation. To illustrate this point, Fig. 2.1 shows simulated delay and energy for a 16-bit multiplier as its supply voltage is scaled. It becomes evident that supply voltage can be scaled down (to about 0.5V in this case) with minimal decrease in delay while providing a large decrease in energy per operation.

At 0.3V an inflection appears in the energy curve. This point is termed the optimal energy point and is the point at which the circuit achieves maximum energy efficiency. This point almost always appears in the sub-threshold region, meaning the supply voltage is below the threshold voltage of the transistors.

Near/Sub-threshold operation differs from super-threshold mainly because in sub-threshold the on-current depends exponentially on threshold voltage (V_{th}) and V_{dd}, while in super-threshold this dependence is linear [2]. In [2] it is shown that

the variation of the on-current can be given by:

$$\frac{\sigma_{I_{sub}}}{\mu_{I_{sub}}} = \sqrt{e^{\left(\frac{\sigma_{V_T}}{n \cdot V_{th}}\right)^2} - 1} \tag{2.1}$$

where n is the sub-threshold swing factor (inversely proportional to V_{dd}), V_T is the thermal voltage, and standard deviation in the threshold voltage (V_{th}) is proportional to $W \cdot L - \frac{1}{2}$. This means there will be more variation in the sub-threshold on-current. Therefore, the delay in sub-threshold circuits will decrease dramatically because of the exponential decrease in the speed of the transistors in this region.

2.3 Variations at Near/Sub-Threshold

Unfortunately, near-threshold operation does come with one major challenge barring widespread adoption. As can be seen in Fig. 2.2 the variability in delays within digital circuits exacerbates wildly as supply voltage is lowered. These delays can vary from chip to chip or even within similar functional units on the same die. These variations come from many sources discussed next in this section. This, on top of the other challenges presented in the rest of this chapter make near-threshold digital logic design much more challenging in practice.

When operating in the near/sub-threshold region, it is important to maximize operating speed while being aware of the increased delay variation (Fig. 3.13). Maximizing the speed will also help to limit the leakage energy of the circuit, mak-

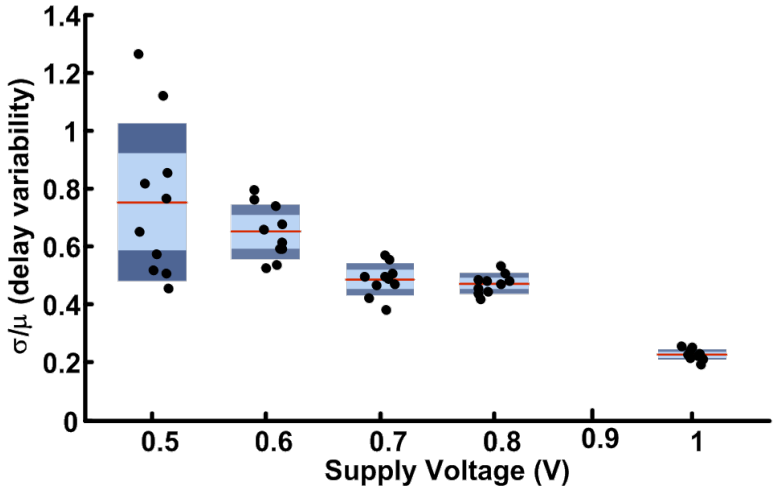


Figure 2.2: Delay variability as supply voltage is scaled to near-threshold for the adder in Fig. 2.1.

ing sub-threshold operation an even better alternative over super-threshold operation with respect to energy consumption. Unfortunately, few methods described in this review can properly operate in sub-threshold. The circuits themselves are also susceptible to increased process variation in near/sub-threshold, which can lead to unreliable operation. These slower speeds lead designers to push sub-threshold processors to a more parallel approach. For example, in [3] the authors use multiple execution lanes to regain the throughput lost by the increased delays of sub-threshold operation.

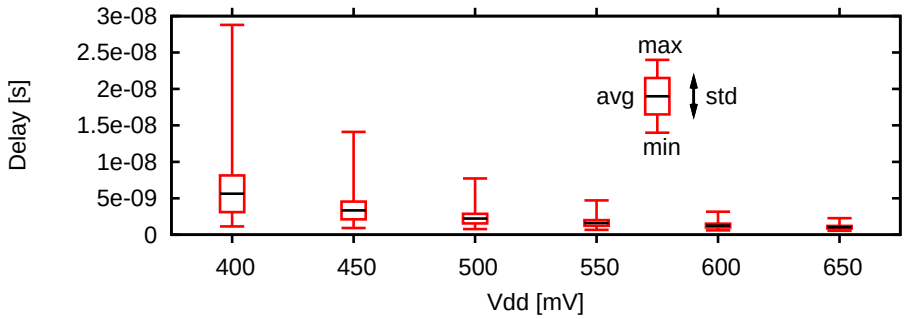


Figure 2.3: Delay variation widens as circuits are driven deeper into the sub-threshold region (Monte-Carlo simulation results from a Multiply-Accumulate in $45nm$ CMOS) [3].

2.4 Sources of Variation in CMOS circuits

There are many sources of variation that plague CMOS circuits today. These variations can be lumped into two categories: static and dynamic. Static variations, primarily process variations [4], do not change over time and typically affect the worst-case path of each die. Dynamic variations, such as changes in temperature [5–8], voltage [5–8], and aging [9], change over time and may require in-situ methods to combat degradation of performance.

2.4.1 Static Variations

Static variations in fabricated dies result in degradation of maximum frequency and increased power consumption. These variations are more problematic in sub-$100nm$ technologies and get worse as process nodes get smaller. Due to the local and global process variation effects in deep sub-micron technologies, the speed of the transistors can vary dramatically from die-to-die or device-to-device.

2.4.2 Dynamic Variations

Dynamic variations are usually categorized by the way they vary over time. Some sources of variation such as supply voltage fluctuations may vary frequently (on the order of a few clock cycles) [10] while others such as aging vary over large periods of time (such as several years) [9]. Because of their dynamic and difficult to predict nature, it becomes much more challenging to design resilient systems to combat these variations.

2.5 Characteristics of Variations

It is important to note that different variation sources have different characteristics that can affect which are used to combat them. The two primary classifications of variations are random and systematic.

2.5.1 Random Variations

Random variations such as line edge roughness effects, and voltage or random dopant fluctuations are not predictable and can result in a wide range of unpredictable variations. Furthermore, random variations can be categorized as both static and dynamic. An example of static random variations is that of random dopant fluctuations which affects transistor threshold voltage V_{th}. An example of dynamic random variations would be voltage fluctuations. For example, we usually do not know which way the voltage will change, and even if we do know how it will

change over time it is hard to say how it will precisely effect a particular circuit.

2.5.2 Systematic Variations

Systematic variations, those due to variations in input operands or photo-lithographic and etching uncertainties [11] always result in predictable change. These variations can be easily characterized and modeled as they are well understood and predictable. For example, it is well known that higher temperatures will result in slower CMOS performance.

2.6 Soft Errors

Soft Errors are a classification of errors whereby after an error occurs it has the possibility of not reoccurring. There are many environmental and design factors that can have an effect on the occurrence of soft errors. This section will list each of these factors and describe how soft errors can be manifested by them.

2.6.1 Temperature-Induced Soft Errors

Soft errors can be generated when the temperature of a CMOS chip gets too high. The temperature relationship to transistor threshold voltage and subsequent delay is well studied [12]. This relationship is shown for clarification in the following equations:

$$V_{th} = V_{t0} + \gamma \left[\sqrt{2\phi_f + V_{SB}} - \sqrt{2\phi_f} \right] \tag{2.2}$$

where

$$\gamma = \frac{\sqrt{2qN_A\epsilon_{Si}}}{C_{ox}} \tag{2.3}$$

and

$$\phi_f = \frac{kT}{q} ln(\frac{N_A}{n_i}) \tag{2.4}$$

As temperature goes up, (usually due to high switching current from high processor utilization) transistors will slow down. As they slow, if a critical path is exercised that has an affected transistor, the latency of that path may result in a timing soft error. This class of soft errors can be prevented with better cooling systems. Also, if allowable, the speed or activity of a processor can be reduced to lower the temperature back down.

2.6.2 Voltage-Induced Soft Errors

Voltage-induced soft errors occur similar in nature to temperature-induced soft errors. There are many ways supply voltage changes (either coming into or within a CMOS die) can lead to a soft error. One such way could be power supply imperfections, such as a decoupling capacitor that is too small resulting in ripple or an unstable supply. Another way could be L(dI/dt) noise from switching within

the die resulting in periodic voltage droops. Either way, as the supply voltage is lowered for any amount of time, transistors can slow down as speed is proportional to voltage. As they slow, if a critical path is exercised (just like in temperature-induced errors) the delay of that path may result in a timing soft error. In many cases, voltage-induced soft errors cannot be directly fixed in-situ. Voltage droops occurring on die are very difficult to correct. However, in some situations it may be possible to raise the voltage higher so that if a droop does happen the droop only goes down to the minimum operating voltage, not below it. This may correct the error at the cost of higher power and complexity.

2.6.3 Process Variation-Induced Soft Errors

Process variations, such as changes in gate oxide thickness, random dopant fluctuation, device geometry effects, and threshold voltage imperfections [4] are essentially imperfections in the microchip manufacturing process. As mentioned in the previous section they are static in nature. It could be argued that process variations do not lead to soft errors as they are not dynamic. However, for the purposes of this dissertation it can be understood that these variations can lead to soft errors in the following way: If a circuit has some amount of process variations applied to its transistors those transistors have some probability of either being slower or faster. If the former is true it could be possible that some circuit designed for $100MHz$ operation can only perform at $90MHz$. This is then considered a soft error because the circuit will continue to operate properly as operating conditions

change. For example, a simple decrease in frequency will fix the soft error. Another option could be to raise the supply voltage in hopes of speeding up the affected transistors.

2.6.4 Aging Effects on Soft Errors

Aging [9] in CMOS circuits can occur by many mechanisms. The principal effect is typically measured as a transistor threshold voltage change over the course of several months to several years. These threshold voltage shifts have a negative impact on speed. Therefore, aging-induced soft errors, for the sake of this dissertation, are considered similar in nature to that of process variation-induced soft errors. Similar methods can be employed to adapt for these errors such as lowering frequency or raising voltage.

2.6.5 Radiation-Induced Soft Errors

A single event transient (SET) or upset (SEU) is a physical phenomenon that occurs when a high-energy particle interacts with the diffusion regions of a MOS transistor and releases charge, potentially causing a faulty transition in a circuit. Although memories in current supercomputers (and other large systems) are often protected against SEUs, logic is often left unprotected. For decades, SETs in logic were a concern only for systems that operate at high altitudes or for which extreme reliability is necessary. As process scaling continues, however, the combination of

smaller devices, lower supply voltage, and increased integration is projected to make single event effects, even in combinational logic circuits, a concern at any system scale [13, 14]. The primary concern with logic SETs is the risk of silent data corruption (SDC), where the application unknowingly produces an incorrect result. This is becoming significant, even if the absolute SER (Soft Error Rate) per processor is low. Therefore, emphasis must be placed on efficient error detection that eliminates this risk without undue overhead. SETs cannot be corrected for as easily as any other type of soft error.

2.7 Hard Errors

A hard error is an error that, no matter what environmental or system change the circuit experiences, always results in an incorrect result [15]. These are typically a result of process variations but can in some extreme environments be radiation induced [16]. Hard errors are an important research topic but are not the emphasis of this dissertation.

2.8 Summary

This chapter introduced and classified the different types of variations in modern CMOS technologies. A detailed description of each type of variations and its impact on error generation was presented. Table 2.1 shows an overview of each of the potential error sources, their properties, and what type of error they result in.

This is intended as an easy reference and the summary of this chapter.

Table 2.1: Classification of variations in digital logic.

	Dynamic		Static		Error Result	
	Random	Systematic	Random	Systematic	Soft Errors	Hard Errors
Temperature	✓	✓		✓	✓	
Voltage	✓	✓		✓	✓	
Process			✓		✓	✓[a]
Aging		✓			✓	✓[a]
Radiation	✓				✓	✓[b]

[a] Marked as soft error sources as they can be corrected if they are just timing errors.

[b] In very high radiation environments permanent failure can occur.

Chapter 3: The Current Art in Circuit Reliability

3.1 Introduction

The growing demand for higher speed and more energy-efficient electronics has forced IC designers to put considerable effort into decreasing the delay and energy consumption of VLSI systems using different circuit and architectural techniques. Although the widely-used technique of pipelining is reliable and effective, it is not flexible and the clock frequency is always limited to the critical path of the system. Some techniques add more flexibility to the design and bypass the critical path of the system, increasing performance. Other circuit techniques aim to reduce the added margins to the clock frequency due to process, voltage, and thermal variations etc.

This chapter introduces the performance improvement techniques proposed in prior research. These techniques are termed speculative speedup techniques as they make a best guess at what improvements can be made to a circuit using only data given to them at the beginning of each clock cycle such as operands and temperature.

As a side note, two different classifications of techniques have been discussed

in the literature. Speculative [17,18], which on the circuit-level, make a best guess at what improvements can be made while tolerating errors; and non-speculative [19, 20], which can be used to improve the reliability of a system without the concern of having to correct for errors, often by using information directly from the data path to potentially make corrections to voltage or frequency margining before errors are detected. However, due to the scope of this dissertation only speculative methods will be discussed.

In Section 3.3 error-recovery techniques that correct a pipelined instruction's operation by either stalling or re-executing it are explained. Next, Section 3.4 talks about higher level error-protection techniques including both architectural and algorithm-level methods. Finally, Section 3.6 discusses the challenges of different techniques under the extreme process variation effects of sub/near-threshold operation.

3.2 Speculative Speed-Up and Error Detection Techniques

Speculative methods usually add a small area and power overhead, but due to their speculative nature they often must confirm their guess and can sometimes suffer a delay penalty if they speculated incorrectly. In an application where the speculative circuit is usually correct, speed-up or energy reduction is achieved. All speculative methods must be paired with some type of error recovery method to ensure the correct operation of a circuit in the presence of errors. These error recovery methods are described in Section 3.3.

As described in the previous chapter, variations can be classified into four modes, dynamic/random, dynamic/systematic, static/random, and static/systematic. Table 3.1 summarizes the applicability of the error detection techniques discussed later in this chapter for each of these categories.

3.2.1 Architectural Retiming

Architectural Retiming gets its name because it both reschedules operations in time and modifies the structure of the circuit to preserve its functionality. It works by eliminating the latency introduced into a circuit by pipeline registers using the concept of a negative register (Fig. 3.2.1). A negative register is an architectural device that contains correct information before the completion of the logic itself. This can be implemented using pre-computation or prediction [21]. With pre-computation, the negative register is synthesized as a function that pre-computes the input to the normal pipeline register using signals from previous pipeline stages. With prediction, the negative register's output is predicted one clock cycle before the arrival of its input using a finite state machine. When a negative register is paired with a normal register the result is architecturally just a wire and does not slow the critical path (assuming predictions are correct).

Architectural Retiming allows for a circuit to continue computation, rather than waiting for the end of a clock cycle, when the result of a pipeline stage finishes early. Pre-computation generates a bypass, an architectural transformation that increases the circuit performance, but it can only do so if there is enough information in

Table 3.1: Ability of different detection methods to work in various variation modes.

	Dynamic		Static	
	Random	Systematic	Random	Systematic
Architectural Retiming				✓
Circuit-Level Speculation[a]				✓
Tunable Replica Circuits[a]	✓	✓	✓	✓
Razor Flip-Flops / Transition Detectors	✓	✓	✓	✓
Architecture Level Error-Protection Techniques	✓	✓[b]	✓	✓[b]
Architectural Data Protection	✓	✓	✓	✓
Software Level Error-Protection Techniques	✓	✓[c]	✓	✓[c]

[a] These methods are designed for dynamic error detection but still operate under static error cases.

[b] Only Simplified Redundancy method can partially handle static variability induced errors.

[c] Excluding Instruction Replication Techniques.

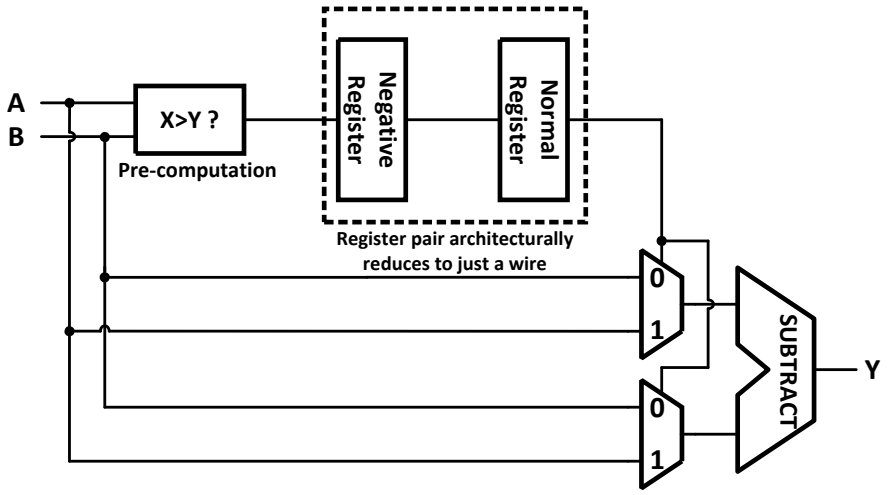


Figure 3.1: Architectural Retiming applied to an subtraction unit, the use of a negative register and regular register simplify to a wire and do not slow the datapath because the negative register stores a decision made in the previous pipeline stage.

the circuit to allow for pre-computing the value one cycle ahead of time. If the information is not available, the circuit can rely on an oracle to predict the value instead, these are usually synthesized with some sort of idea of what the logic will be doing and can be application specific. If a prediction is performed, it is necessary to verify the predicted value one cycle after the prediction when the actual value is computed. If the prediction is correct, the system can proceed with the next prediction. If the prediction was incorrect, the mis-predicted value must be flushed and the circuit should be restored to the previous state. This results in at least one cycle latency penalty. The applications of this technique are limited and the area and power overhead of the pre-computation or the prediction circuit is high.

3.2.2 Circuit-Level Speculation

Circuit-Level Speculation [18] is a method introduced to reduce the critical path of the circuit using approximation, implementing only a portion of a circuit. The approximated implementation of the circuit is the most heavily used part of the original circuit based on simulation results. A redundant full circuit is implemented and works in parallel to verify the approximated result in the next cycle (Fig. 3.2.2). Although this method results in a potentially large speed-up, it also requires a larger area and thus more power consumption. For example, when implementing a partial adder in the critical path, a complete adder has to work in parallel as well as an additional adder to compare the results in the next cycle. The energy overhead of this technique makes it unfeasible for use in low-power applications. The circuit may also suffer a delay penalty if the approximated result does not match the complete result.

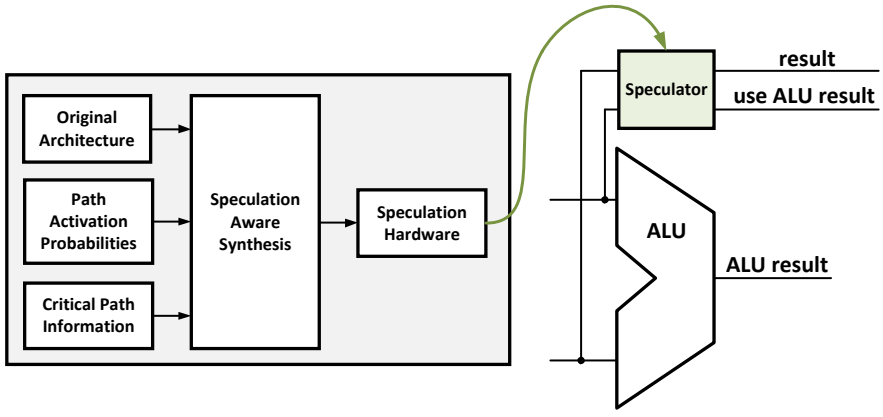


Figure 3.2: An example of a typical circuit-level speculation scheme: a speculator is used to speedup the system based on path activation probabilities from a model.

3.2.3 Tunable Replica Circuits

One method for finding errors due to voltage and frequency changes is the use of Tunable Replica Circuits (TRCs) [10]. These circuits are composed of a number of digital cells, such as inverters, NAND, NOR, adders, and metal wires that are tunable to a given delay time (Fig. 3.3). The replicas are affected by variations in a similar way to the critical path. Once the replica is tuned to the critical path, it will replicate the path delay as it changes due to variations. The TRCs can be used to report the critical path delay using a thermometer code or perform dynamic error detection.

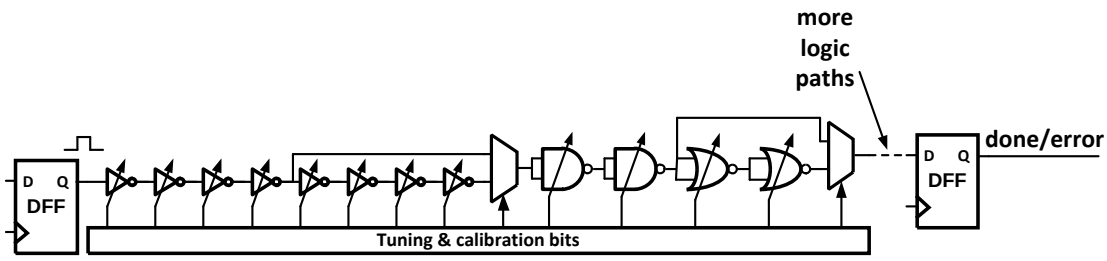


Figure 3.3: A Typical Tunable Replica Circuit (TRC).

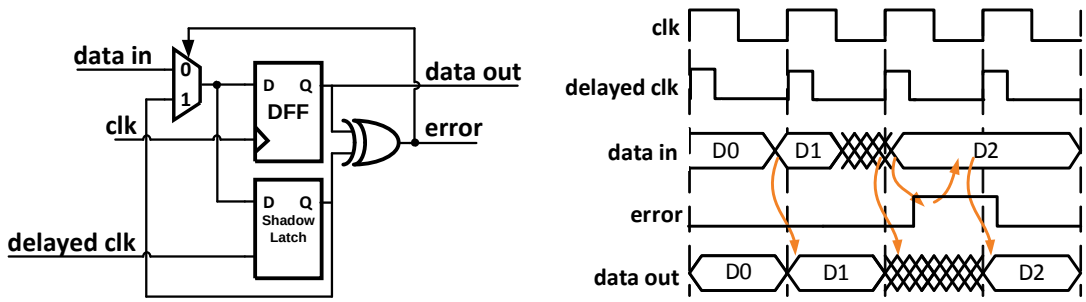
TRCs are able to detect errors without introducing additional components or time delays to the data path. After being tuned once, the TRCs will mirror slightly worse than the critical delay path to ensure that any timing violation will be observed.

TRCs do suffer from a few drawbacks. Because the circuit is only a worst-case replica, it is possible that they will trigger error responses when there was not a timing violation in the actual data path. The circuits themselves also take

up area and power. Finally, TRCs cannot adapt to unique delay paths, only a simple worst-case. This can make them hard to calibrate correctly under extreme variations as the TRC's timing margin needs to be large enough to guarantee all errors will be caught correctly. This also limits the speed-up potential while using TRCs.

3.2.4 Razor Flip-Flops

Razor [22] works by pairing each flip-flop within the data path with a shadow latch which is controlled by a delayed clock. As shown in Fig. 3.4, after the data propagates through the shadow latch the output of both of the blocks is XOR'd together. If the combinational logic meets the setup time of the flip-flop, the correct data is latched in both the data path flip-flop and the shadow latch and no error signal is set. Different values in the flip-flop and shadow latch indicates an erroneous result was propagated, and when an error is detected a logic-high



(a) (b)

Figure 3.4: (a) Synthesizable Razor error detector (b) Example timing diagram of error detection.

signal is broadcast to an error recovery circuit as described in Section 3.3. The possibility exists that the datapath flip-flop could become metastable if setup or hold-time violations occur. Razor uses extra circuitry to determine if the flip-flop is metastable. If so, it is treated as an error and appropriately corrected.

Razor II [23] uses a positive level-sensitive latch combined with a transition detector to perform error detection Fig. 3.5. Errors are detected by monitoring transitions at the output of the latch during the high clock phase. If a data transition occurs during the high clock phase, the transition detector uses a series of inverters combined with transmission gates to generate a series of pulses that serve as the inputs to a dynamic OR gate. If the data arrives past the setup time of the latch, the detection clock discharges the output node and an error is flagged. Replacing the datapath flip-flop from Razor with a level-sensitive latch eliminates the need for metastability detection circuitry. By removing the master-slave flip-flop and metastability detector, this version shows improved power and area over Razor.

When using Razor, it is important to be aware of the trade-offs that exist in achieving correct utilization of the timing window that enables Razor to properly detect errors. If a short path exists in the combinational logic and reaches the error latch before the delayed clock-edge of the computation preceding it, a false error signal could occur. To correct this, buffers are inserted in the fast paths to ensure that all paths can still be correctly caught. While this can help to guarantee a minimum timing constraint (hold time) of the shadow latch is met, it will also lead to additional area and power.

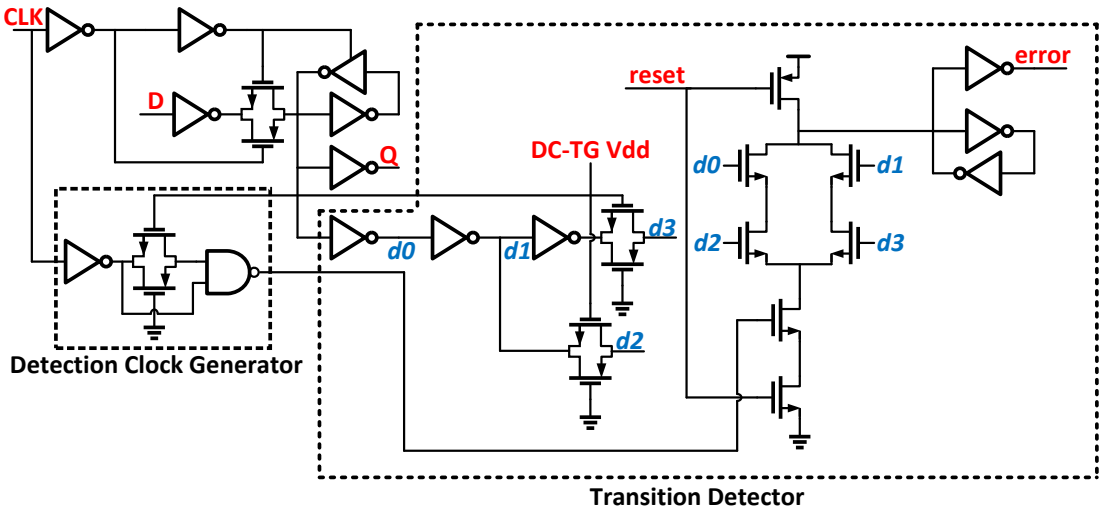


Figure 3.5: Razor II latch with detection clock generator and transition detector.

3.2.5 Transition Detectors

A number of other methods exist that also serve as capable error detection methods. The transition detector with time-borrowing (TDTB) [7] is similar to Razor II in that it uses a dynamic gate to sense transitions at the output of a level-sensitive latch. Shown in Fig. 3.6(a), the TDTB differs from Razor II by using an XOR gate to generate the detection clock pulse, and the dynamic gate used in the transition detector uses fewer transistors.

Double sampling with time-borrowing (DSTB) [7] is similar to the previous circuit but with the transition detection portion replaced with a shadow flip-flop (Fig. 3.6(b)). Like Razor, the DSTB double samples the input data and compares the data path latch and shadow flip-flop to generate an error signal. The advantages of DSTB are that it also eliminates the metastability problem with Razor

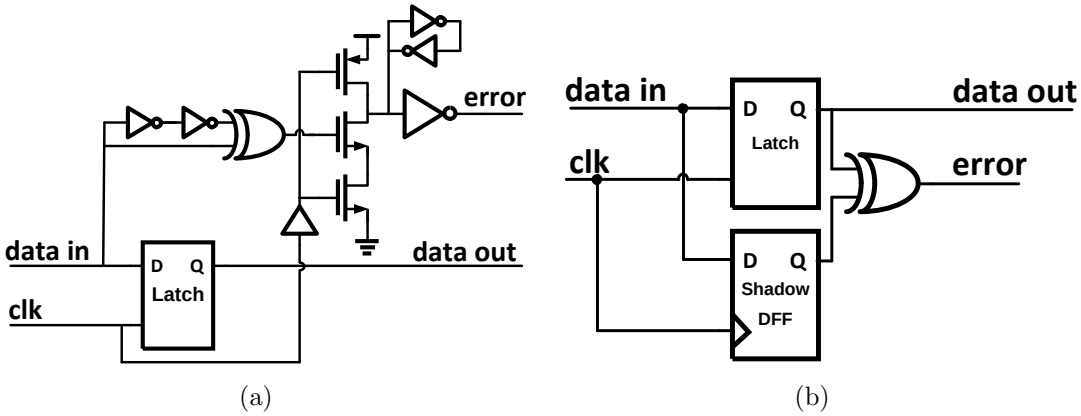


Figure 3.6: Two versions of modified Razor flip-flops: (a) Transition Detection with Time Borrowing (TDTB). (b) Double Sampling with Time Borrowing (DSTB).

by having the flip-flop in the error path and retaining the time-borrowing feature from the transition detector. Clock energy overhead is lower than Razor since the data path latch is sized smaller than the flip-flop used in Razor. Aside from this, DSTB retains similar issues to Razor.

The static and dynamic stability checkers are introduced in [24]. In the static stability checker, the data is again monitored during the high clock phase using a sequence of logic gates. If the input data transitions at all during the high clock phase, an error signal is generated. The dynamic stability checker uses a series of three inverters to discharge a dynamic node in the event of a data transition during the high clock phase, generating an error signal. In [25] soft error tolerance is achieved by using a combination of time and hardware redundancy. This results in less hardware being used, but increases the time needed to check for errors.

3.3 Error Recovery Techniques

Once an error has been detected, a method needs to be in place to allow for the error to be dealt with properly. In a pipeline, later instructions may depend on the data generated by an earlier, errant instruction. Therefore these methods need to both assure the error is fixed (either by waiting enough time for the error to be corrected or re-executing the errant instruction) and make sure the erroneous instruction does not propagate the error.

Many pipeline error recovery techniques have been explored in the past. The most researched of these techniques are: Clock Gating [5,22], Counterflow Pipelining [22], Micro-Rollback [22], and Multiple Issue [6]. The following section discusses each approach and their advantages and drawbacks.

3.3.1 Clock Gating

Clock Gating is conceptually the simplest technique to implement of all error recovery methods. Its original purpose was for saving power of unused blocks on a systems level by not clocking them when they are not used. This technique can also be adapted to error recovery by pausing all pipeline stages while waiting for the slow stage to either finish computation or to allow for the instruction to be re-executed. The pausing action ensures that later instructions do not continue to their next pipeline stage until the errant instruction is corrected. It is most commonly paired with Razor flip-flops as it only works if the pipeline can be stalled before the next clock edge, before the pipeline registers are set to get new data

which can be achieved in slow systems. The Clock Gating concept is illustrated in Fig. 3.7.

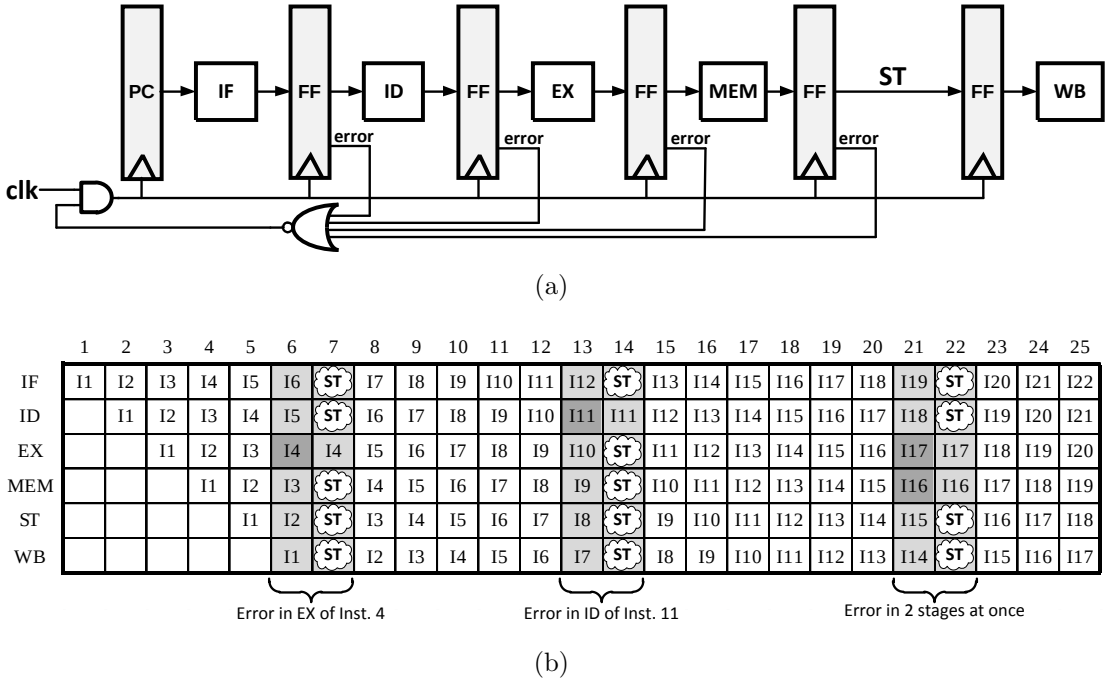


Figure 3.7: (a) Pipeline modification for Clock Gating error recovery method. (b) Clock Gating pipeline data path with errors.

The primary advantage to this method is that it requires very little architectural changes as well as minimal area addition to a design compared with other methods. However, in order for this method to work properly, a stall signal needs to propagate to all pipeline stages in a very short amount of time (50% of one clock cycle when Razor circuits are used). This can be difficult to achieve across large CMOS dies where pipeline stages are several millimeters apart. Furthermore, this is completely impractical to implement in complicated microprocessors because it may take several clock cycles just to propagate the clock signal through a clock

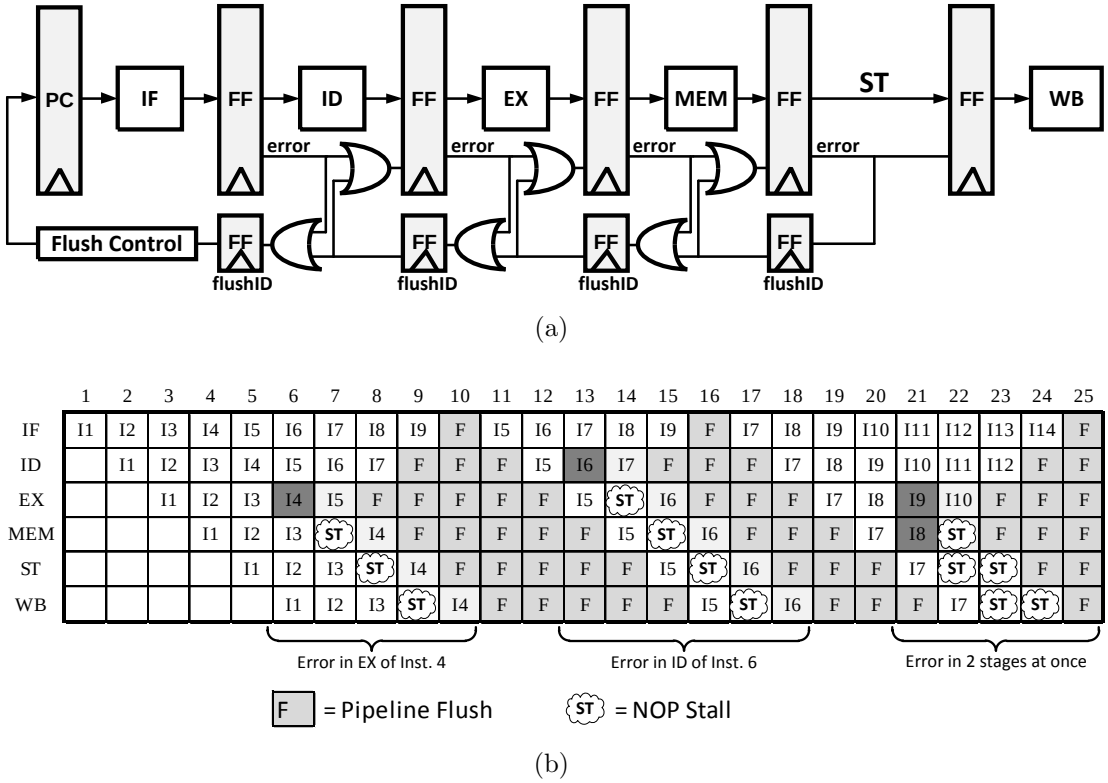


Figure 3.8: (a) Pipeline modification for Counterflow Pipelining error recovery method. (b) Counterflow pipeline data path with errors.

distribution network which cannot be halted in only one cycle.

3.3.2 Counterflow Pipelining

Traditional Counterflow Pipelining is a microarchitecture technique that uses a bidirectional pipeline, allowing instructions to flow forward and results to flow backward. This technique made it easier to implement operand forwarding, register renaming, and most importantly, pipeline flushing [22,26]. In order to modify this

technique for error recovery, a traditional pipeline is modified such that only the flush signals are bi-directional. This concept is depicted in Fig. 3.8. In the event of an error, flush registers begin to propagate the error signal until it reaches the flush control unit. At that point the Program Counter (PC) is updated with a corrected instruction pointer and the pipeline continues operation. Because the flush registers clear the pipeline in both directions as the error is propagated, there is no need to do anything other than resume execution after the error has finished propagating. An illustration the Counterflow Pipeline instruction flow can be found in Fig. 3.8(b).

This method only requires local information to determine a stall, there is no global stall signal that needs to be computed and transmitted such as in clock gating. However, depending on how this method is implemented the area/power overhead can get quite large. For example, there needs to be several PC values stored in the Flush control unit or in the flushID registers themselves; this overhead can get large in a 64-bit CPU with high pipeline depth. Unlike Clock Gating, this method takes several more cycles to recover from an error as the error propagates back one stage per cycle and the pipeline needs to be flushed.

3.3.3 Micro-Rollback

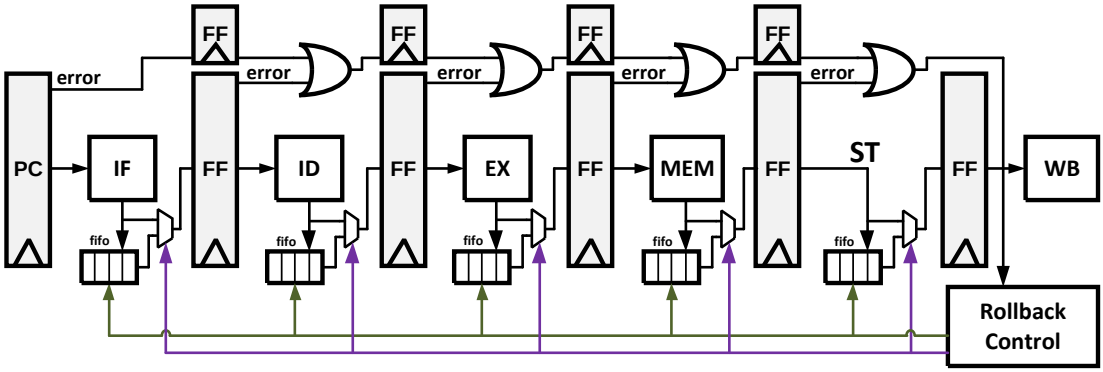
Micro-Rollback is a technique that saves a queue of previous instructions and operands at each pipeline stage [27]. After a successful operation, each stage saves the results as they are passed to the next stage. If an error is detected, instructions

can be restarted at their last known correct state in the pipeline. Once an error is detected a signal is propagated with the instruction as it continues through the pipeline. Once the instruction reaches the write-back stage, rollback logic is activated to stop the instruction from being written and sends a signal to all of the pipeline stages to roll back and retry the instruction. The Micro-Rollback architecture is illustrated in Fig. 3.9(a). In its original design, instructions were simply replayed with the hope of the error being resolved at a later time. This is not very robust but with small architectural modifications this method can be redesigned to replay instructions twice or three times to ensure no error is produced a second time.

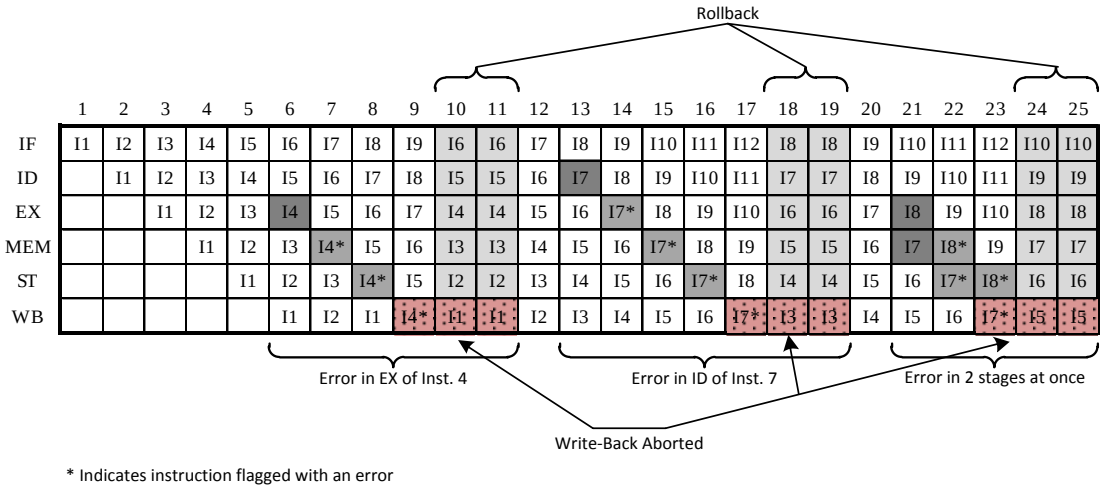
Using this technique makes the recovery process several cycles long. However, it reduces the error propagation/control overhead at each pipeline stage as required in the previous two techniques, leading to a faster clock speed. Unfortunately because of the addition of the queues, energy consumption may grow by 15% or more [22].

3.3.4 Multiple Issue

This method, shown in Fig. 3.10 has the simplest architecture of all the correction schemes. Similar to Micro-Rollback, errors propagate to the write-back stage, but instead of rolling back to a specific state, the entire pipeline is flushed and the erroneous instruction is replayed multiple times in succession to ensure completion of the previously failed operation. The amount of replayed instructions can vary de-



(a)



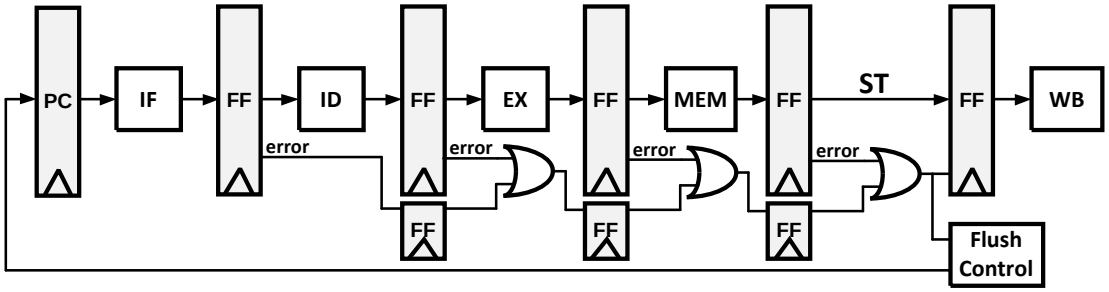
(b)

Figure 3.9: (a) Pipeline modification for Micro-Rollback error recovery method. (b) Micro-Rollback pipeline data path with errors.

pending on the expected delay of the erroneous stage. For example, in Fig. 3.10(b) the instruction is replayed three times. This is the most popular method proposed to be used in systems where corrected data cannot be captured after an execution cycle (such as in TRCs).

One advantage to this method is that it requires very small overhead in control

logic which results in less area and power. There is however a significant delay when a failed instruction is found as the pipeline has to be flushed and the instruction replayed three times. Depending on the frequency of errors this may result in a large average energy overhead per instruction as they take up many more clock cycles to complete. However, if these errors happen infrequently there is minimal penalty.



(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
IF	I1	I2	I3	I4	I5	I6	I7	I8	I9	F	F	F	F	F	F	I4	I4	I4	I5	I6	I7	I8	I9	I10	I11
ID		I1	I2	I3	I4	I5	I6	I7	I8	I9	F	F	F	F	F	F	I4	I4	I4	I5	I6	I7	I8	I9	I10
EX			I1	I2	I3	I4	I4	I5	I6	I7	I8	F	F	F	F	F	F	I4	I4	I4	I5	I6	I7	I8	I9
MEM				I1	I2	I3	I4*	I5	I6	I7	I8	I9	F	F	F	F	F	F	I4	I4	I4	I5	I6	I7	I8
ST					I1	I2	I3	I4*	I5	I6	I7	I8	I8	F	F	F	F	F	F	I4	I4	I4	I5	I6	I7
WB						I1	I2	I3	I4*	I5	I6	I7	I7	I8	F	F	F	F	F	F	I4	I4	I4	I5	I6

Error in EX of Inst. 4 | Invalid Instructions | Pipeline Flush | Replicas | Valid Instructions

(b)

Figure 3.10: (a) Pipeline modification for Multiple Issue error recovery method. (b) Multiple Issue pipeline data path with errors.

3.3.5 Adaptive Scaling Methods

Once an error (or errors) has been detected, it may be beneficial to decrease the clock frequency or increase V_{dd} to ensure that the frequency of errors decreases in the future. Adaptive scaling methods such as Dynamic Variation Monitors (DVM) [28, 29] usually combine some type of error detector such as TRCs or RAZOR circuits with time-to-digital converters, error counters, or some type of prediction logic. If it has been determined that errors are occurring too frequently, different circuit parameters such as V_{dd}, core frequency, and clock skew are scaled accordingly. In the case of dynamic V_{dd} droops, this scaling can happen at a finer time-granularity. Courser tuning for slower changing variations such as instruction/data or temperature dependent delays can also be achieved. Fig. 3.11 illustrates this feedback process on the systems level.

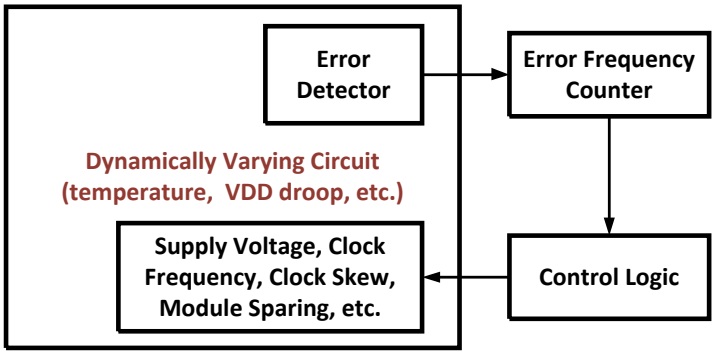


Figure 3.11: A systems-level diagram illustrating the feedback process of dynamically adapting circuit performance in the presence of dynamic variations.

3.3.6 Checkpoint-Restart

Checkpoint-restart (or checkpoint-rollback) is a common recovery technique that can be based entirely in software. System state is occasionally preserved in a checkpoint. When an error is detected, the system state is rolled back to the most recent checkpoint and execution is restarted. Checkpoint-restart is traditionally a software technique with high overhead. Because of this its applicability to very low error rates is limited. Researchers proposed that the hardware structures of processors that support speculative execution can also be used for low-overhead checkpoint-restart [30, 31]. Even with speculation hardware support, the overhead is still tens of cycles requiring a fairly low error rate to be effective.

3.3.7 Summary of Recovery Methods

Table 3.2 shows the advantages and disadvantages for each recovery method. It is clear that certain methods may prove to be better than others depending on the requirements of the pipeline and other system constraints. As far as complexity is concerned it is clear that the Multiple Issue method is the highest performer, as Clock Gating is almost never practical in large systems. However, in a system where errors are more frequent and max clock speed is a concern, methods such as Counterflow Pipelining may be more beneficial.

Table 3.2: Comparison of Pipeline Recovery Methods.

	Area	**Power**	**Complexity**	**Recovery Time**[a]
Clock Gating	High	High	Impractical	1 cycle
Counterflow	Medium	Medium	Medium	1+(N-1) cycles
Micro-Rollback	High	High	High	N cycles
Multiple Issue	Low	Low	Low	2N+2 cycles

[a] worst-case recovery time (number of cycles after fault before next instruction) where N = pipeline depth

3.4 Higher Level Error-Protection Techniques

Although this chapter's main focus is on circuit level techniques, in this section a survey of mechanisms in architecture and algorithm levels is also presented. Using these higher level approaches usually will not rely on error protection in levels beneath. Thus, it will allow achieving error protection while using commodity (non-protected) components. For example, architectural level techniques will provide error protection while using non-protected circuits, and algorithm level approaches can offer error protection while operating on commodity hardware.

Some if not all of the approaches described in this section have been initially designed in the context of system fault tolerance. As such, one of the basic assumptions of these approaches is fault randomness and independence. It makes these approaches suitable to handle random variation, however, limits their effectiveness in handling systematic variation.

3.4.1 Architecture Level Error-Protection Techniques

All architectural error protection approaches rely on some form of replication. Some techniques are using exact replication and therefore are more generic, while others provide error protection using simplified duplicated instances.

3.4.1.1 Dual Modular Redundancy

One of the simplest forms of architectural level error-protection is Dual Modular Redundancy (DMR). The protected module is replicated such that both the original module and its copy share their input signals (Fig. 3.12(a)). Outputs are compared and a mismatch indicates an error. In case of an error, the system has to restore its last verified state and re-execute from that point. While having the advantage of design simplicity in using exact replication, this approach's disadvantages include high area and power/energy overheads. In addition it requires a roll-back and replay mechanism to recover in case of an error.

3.4.1.2 Triple Modular Redundancy

Triple Modular Redundancy (TMR) is similar to DMR, but this approach utilizes two replicated instances in addition to the original module (Fig. 3.12(b)). The outputs of the three instances are compared and a majority voter mechanism is used to select the outcome signals. Based on the assumption that only one of the instances (at most) will fail at any given point, this approach eliminates the need

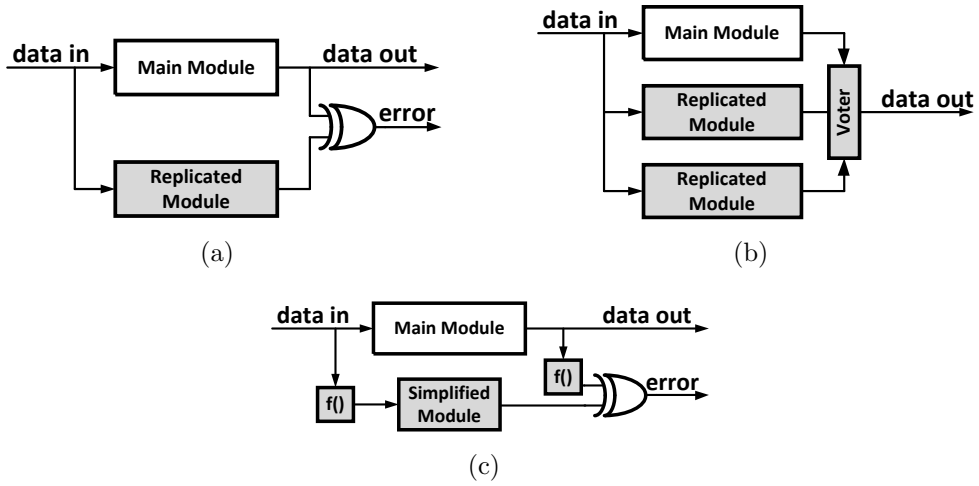


Figure 3.12: Architecture Level Error-Protection Techniques. (a) Dual Modular Redundancy (DMR). (b) Triple Modular Redundancy (TMR). (c) Simplified Redundancy.

for a roll-back and replay mechanism used in DMR, allowing instant recovery. One disadvantage is that it increases the area and power/energy overheads by adding the third instance of the module as well as increasing the critical path by adding the voter mechanism.

A generalized version of TMR is N-Modular Redundancy (NMR) also known as M of N redundancy. In this configuration, N instances of the module process the same inputs in parallel, and then use a majority voting scheme where M is the minimum number of modules required to get a majority count out of N modules. To summarize, NMR introduces xN power/area overhead and can tolerate at most N-M modules to fail.

3.4.1.3 Simplified Redundancy

To avoid the high overhead of the replicated module introduced in NMR schemes, it is possible to use a simplified version of the original module instead (Fig. 3.12(c)). Several techniques based on this approach have been proposed. In [32], the researchers propose using a simple (inorder) core as a checker for a complex control-intensive superscalar processor. The authors show that using a simple core as a checker significantly reduces the overhead compared to DMR while providing reasonable coverage.

A simplified checker approach can be used with arithmetic circuits as well. The key is to check the arithmetic operation with a similar operation of reduced bit-width. This is possible by transforming the operands into a reduced space with a transformation that is preserved under the arithmetical operations (+, -, *, /). A common transformation used for this purpose is the residue code [33]. The residue code works by transforming the operands, by calculating the remainder of their division by a constant A. Thus the outcome of the main arithmetic circuit can be verified by comparing its transformation (remainder of their division by a constant A in case of residue code) with the result of the checker circuit (Fig. 3.12(c)). Although calculating a remainder can be a complicated operation, using 2n-1 as the constant A allows for calculating the reminder using simple logic and avoiding the expensive division. Using A=3 will protect from all single bit flip errors, but might not detect some multiple bit flips. Error coverage can be increased using larger values of A.

Although the residue code is applicable to integer arithmetic only, there are some approaches to use it for floating point arithmetic by protecting various stages of floating point computations separately [34]. In addition, other codes can be used in the context of floating point, like Berger codes [35]. The disadvantage of Berger codes, however, is their lower coverage.

3.4.2 Architectural Data Protection

The same principles or redundancy can be applied in the context of data transport using, for example, buses or crossbars, or to storage structures. While it is possible to transport (or store) multiple copies of the same data to provide error protection, it is also possible to use a simplified (reduced) copy of the original data for protection purposes.

The simplest approach is known as a parity bit. By storing or transporting an additional bit whose value is equal to a XOR of all the bits of the data, a single bit flip error can be detected. This approach introduces very low overhead of a single additional bit. However, it can detect only a single bit flip and may not detect multiple bit errors.

While the parity bit approach provides a method for error detection only, the error-correcting codes (ECC) approach allows for the correction of the detected error, achieved by increasing the amount of redundant bits. Various approaches are available in the literature, varying the amount of error coverage and the overhead (Table 3.3 on Page 43). The decoding and encoding logic for the simple codes

shown in Table 3.3 is negligible compared to the overhead of redundant storage and bandwidth [36].

3.4.3 Software Level Error-Protection Techniques

Maintaining some level of error-protection while using non-protected hardware can be achieved using software level error-protection techniques. Being applied at the high level of software, these approaches are the most flexible in terms of the trade-off between the error coverage and the overhead. However, these techniques are very limited in the amount of coverage they can provide.

3.4.3.1 Assertion Based Error-Protection

A very efficient way to detect errors in computation is to add assertions and invariant checks based on algorithmic knowledge [37–39]. This approach can be effective in some cases [40], however, the error coverage that can be achieved in the general case is relatively low. In addition, assertions often require expert knowledge and utilize algorithm-specific traits, both of which are not always available.

3.4.3.2 Algorithmic Based Fault Tolerance (ABFT)

Using a modified version of the algorithm that operates on redundancy encoded data to verify the main result can be used for errors detection. In addition, in

specific cases it can also allow for error correction. This approach has relatively low overhead and potentially can provide high error coverage.

Various algorithms implementing ABFT are available in the literature [41–47]. However, the disadvantage is that this technique should be tailored specifically for each algorithm, requiring time-consuming algorithm development. Moreover, this approach is not applicable to an arbitrary program.

3.4.3.3 Arithmetic Codes

Perhaps the simplest example of an arithmetic code is the AN code [33] (also known as linear residue code, product code, and residue-class code) applicable for addition (and subtraction) operations on integers. The operands are encoded by multiplying them by a constant A. Based on the arithmetic properties of addition (subtraction), the result should also be a multiple of A. Thus, it allows validating the result, signaling an error if it is not a multiple of A.

3.4.3.4 Instruction Replication Techniques

Instruction replication, which introduces computational redundancy by duplicating all instructions in software, is arguably the most general software-level technique. While very general and amenable to automation [48–51], instruction replication has potentially high performance and energy overheads. Attempts have been made to minimize this overhead by executing instructions back-to-back in the same func-

tional unit. This approach might be sufficient to detect particle-strike type errors but a different approach is necessary to detect variation-induced errors, such as executing at different times to detect voltage-variation related errors or on different functional units to address static variations.

3.5 Summary Error Detection/Protection of Methods

The last two sections contain a rather dense summary of many of the existing speculative techniques. Table 3.5 was created in order for the reader to attain a more general understanding of the pros and cons of each method and how they relate to reliability.

Table 3.3: Various ECC approaches overheads (based on [52])

	SEC-DED : single bit error correction, double bit error detection [53, 54]		SNC-DND : single nibble error correction, double nibble error detection [55]		DEC-TED : double bit error correction, triple bit error detection [56]	
Data bits	Redundant bits	Overhead	Redundant bits	Overhead	Redundant bits	Overhead
16	6	38%	12	75%	11	69%
32	7	22%	12	38%	13	41%
64	8	13%	14	22%	15	23%
128	9	7%	16	13%	17	13%

Table 3.4: Comparison of existing errors detection/prediction methods

	Reliability Advantages	**Reliability Challenges**
Architectural Retiming	Potential Speedup (31%)	Works for some logic, not all
Circuit-Level Speculation	Potential Speedup (36%-88%)	Only adder is considered
Tunable Replica Circuits	Ensures proper worst-case performance	Many false positives
Razor Flip-Flops / Timing Error Detectors	Detection window limited to $\tilde{2}$0% of clock	Potential for hold-time violations on shadow latch
Architectural Level Error Protection Techniques	Reliable, low design cost	Hard to verify in-situ, large overhead, seldom used
Architectural Data Protection	Potentially low overhead, well understood	Hard to design for logic
Software Level Techniques	Almost no change needed in hardware	Can't trust computer scientists

3.6 Error Detection and Recovery in Near/Sub-Threshold

When operating in the near/sub-threshold region, it is important to maximize operating speed while being aware of the increased delay variation (Fig. 3.13). Maximizing the speed will also help to limit the leakage energy of the circuit, making near/sub-threshold operation an even better alternative over super-threshold operation with respect to energy consumption. Unfortunately, few methods described in this review can properly operate in near/sub-threshold. The circuits themselves are also susceptible to increased process variation in near/sub-threshold, which can lead to unreliable operation. These slower speeds lead designers to push near/sub-threshold processors to a more parallel approach. For example, in [3] the authors use multiple execution lanes to regain the throughput lost by the increased delays of near/sub-threshold operation.

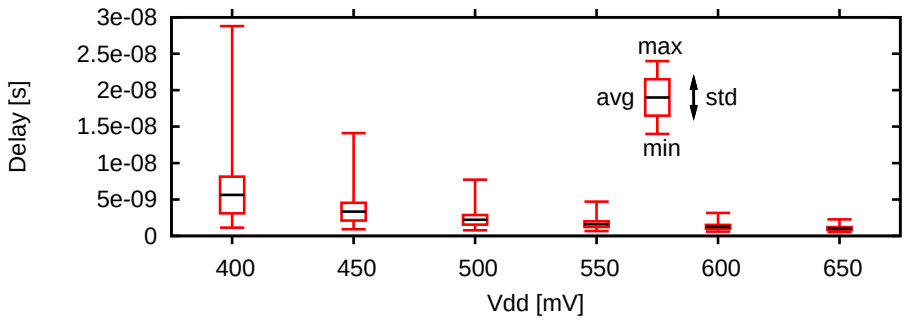


Figure 3.13: Delay variation widens as circuits are driven deeper into the sub-threshold region (Monte-Carlo simulation results from a Multiply-Accumulate in $45nm$ CMOS) [3].

The following subsections will discuss the viability of the techniques under review to operate in the near/sub-threshold region. This section begins by discussing

speculative error detection techniques and continues on with error recovery techniques. Finally, specific techniques that have been designed for the purpose of near/sub-threshold operation will be discussed.

3.6.1 Speculative Error Detection in Near/Sub-Threshold

One of the biggest hurdles facing designers of near/sub-threshold error detectors is the uncertainty of the critical path. When designs are synthesized for super-threshold, accurate timing libraries are used to ensure the critical path is known and can be tested on-die. With the increased variations of near/sub-threshold the critical path may actually change as the voltage is lowered. This characteristic of near/sub-threshold operation makes speculative error detection techniques nearly impossible to operate efficiently.

First off, in order to use a speculative approach, a circuit must be well characterized. This would have to be done on a chip-by-chip basis as delays can change wildly from die-to-die. This process, just for a simple 32-bit multiplier, would take several decades to characterize thoroughly. There may be some promise in using a re-configurable speculative approach if a circuit could be accurately characterized on-die.

Non-speculative error-detection methods have the most promise for working in near/sub-threshold correctly. Methods like Telescopic units require that the critical path is well known and have the same issue as speculative techniques. However, more adaptive methods like TRC circuits have potential to operate

correctly down to sub-threshold. Moving near/sub-threshold circuits to a more asynchronous mode of operation probably has the most promise for the variation tolerance needed in this region of operation. Razor circuits, if designed properly, can operate in near/sub-threshold. Section 3.6.3 discusses an implementation of a near/sub-threshold Razor system.

3.6.2 Error Recovery Techniques in Near/Sub-Threshold

One of the most important factors to use when comparing any technique for sub-threshold operation is the energy overhead required to use it, especially the use of flip-flops as they consume an ill-proportionate amount of energy in sub-threshold. When comparing error-detection methods, there are two options that have minimal energy overheads. The most practical option is Multiple Issue as it has the lowest energy overhead while being a realistic option for robust error recovery. However, just as this method has a small energy/cycle overhead it takes many more cycles to complete the recovery process. Depending on the frequency of errors this method may be completely impractical as the whole pipeline is leaking for 2N+2 clock cycles. In order to determine which error recovery method is truly the best for near/sub-threshold, two factors must be taken into account and static energy overhead versus recovery time should be the metric used. The designer should determine the percentage of errors they expect to tolerate before using this metric as energy per throughput could change dramatically as the frequency of errors is changed.

3.6.3 Near/Sub-Threshold Timing Error Detection

One idea that exists in [57] is to re-design the Transition Detector with Time-Borrowing (TDTB) circuit [7]. By carefully sizing the transistors, they limit the V_{th} variation and obtain the optimal drive strength ratio of NMOS to PMOS for low-V_{dd} operation. They also add extra logic to ensure correct functionality and use high-voltage threshold (hvt) transistors to reduce leakage as flip-flops are one of the largest contributors to sub-threshold leakage. This paper also introduces good method of measuring the effectiveness of an error detection circuit by generating a graph showing the upper and lower frequency limits that errors can successfully be detected at each voltage. As research into near/sub-threshold error detection continues, this will be an important metric to compare error detection methods against each other.

3.7 Conclusions

Variations of all forms must be dealt with in all time-constrained digital systems today. This chapter summarized techniques to improve both the variation tolerance of these circuits as well as their throughput. Speculative techniques make a best guess of a circuit's delay based on known timing to improve throughput and potentially postulate timing induced errors. Once errors have been detected properly, error recovery methods are used to either stall or rollback the pipeline to ensure incorrect instructions are computed correctly. Furthermore, high-level techniques were explored that can improve robustness without the need for robust

circuits.

Robust error detection and recovery can be challenging especially when operating in the near/sub-threshold region. Many of the techniques summarized in this review have great potential for robust operation under extremely variable conditions. The most valuable techniques that will continue to be used in highly variable conditions are: well-designed Razor circuits and recovery methods similar to the Multiple Issue method, paired with higher-level techniques such as error-correcting codes. This of course will be combined with complex systems to control and manage temperature and voltage fluctuations to reduce future errors due to changing conditions.

Chapter 4: A Design Automation Methodology Approach

4.1 Introduction

Some of the techniques presented in the previous chapter have been deemed potentially viable solutions to combating variations in a near/sub-threshold environment. However, problems may exist that don't allow these circuits to operate properly at lower supply voltages. Digital circuits must first be generated from code by complex tools before being placed in a design. This process is known as standard cell synthesis whereby a library of standard cells are used to represent a logical design from code that the designer writes.

Several problems exist that prevent conventional standard cell libraries, characterized at high supply voltages (0.9V-1.2V), from functioning efficiently in the near/sub-threshold region (0.2V-0.6V). The most prevalent problem is leakage current asymmetry [58], where longer delays or even incorrect logic outputs can arise due to the summation of multiple leakage paths, creating a leakage current imbalance. A second issue involves improper ratio inverter feedback in flip-flops and latches that hinder their ability to switch properly with low supply voltages [58]. These problems can result in either reduced circuit speed or complete logic failure

of synthesized logic while running at near/sub-threshold supply levels.

Previous work has dealt with these problems by manually modifying a standard cell library and replacing cells with new cell topologies [59], and [60], re-characterizing the library completely [61], or creating a new library altogether [62]. Unfortunately, solving the problem with the above solutions can be inefficient and time consuming. Currently, designers often do not have the tools needed to alter a standard cell library in a timely manner, as individual cell schematics and layouts need to be re-optimized for a particular process. If the RTL synthesis is implemented in another technology, a different custom standard cell library needs to be recreated again. Finally, functional verification of these new custom cells is difficult and time consuming, as simulating across multiple process corners, low supply voltages, and Monte Carlo variations are essential to ensuring design robustness.

In this chapter, a new design methodology is proposed that aims to improve the functionality and yield of a standard cell library operating in the near/sub-threshold region. This procedure automatically removes standard cells that exhibit poor operating characteristics from the RTL synthesizer.

By removing poorly performing cells, this cell-culling methodology improves several aspects of sub-threshold design:

- Improved yield for higher clock rates, as outlier cases decrease under Monte Carlo simulation.

- Improved DVFS (Dynamic Voltage-Frequency Scaling) [63] capability and robustness, as operation occurs more efficiently over a wider range of supply

voltages.

- Area and delay are decreased, resulting in lower energy per computation.

The outline for this chapter is as follows. First, a discussion of the performance of conventional standard cell libraries in the near/sub-threshold region is presented. Next, the proposed design automation method that culls under-performing cells is presented. The chapter will then conclude with simulated results across Monte-Carlo variations in a 90nm-CMOS technology.

4.2 Limitations of Existing Libraries and Design Techniques in Near/Sub-Threshold

Typically, standard cells are designed to operate optimally at their typical operating supply voltage (i.e. VDD=1.2V), and characterized across process corners such as slow, typical and fast corners. However, when using the library in the sub-threshold region (i.e. VDD=380mV), some cells operate poorly when compared to others. One reason for this undesirable operation is because the Ion/Ioff ratio is degraded [58] as the supply voltage of a cell is lowered. At typical supply voltages, Ion current dominates Ioff, or leakage current. This ratio ensures that logic 1 is easily within 10% of the supply voltage and logic 0 is within 10% of the supply ground, and symmetrical rise and fall times are maintained (if required), providing optimal input-to-output delay. As the supply voltage is lowered, the Ion current weakens much more dramatically than the Ioff current. At this point,

unwanted sub-threshold leakage current begins to dominate and cell performance is degraded. There are many factors that can exacerbate this degradation such as transistor sizing [64], transistor stacking [59], and poorly designed cells [59]. The following will describe why and where synthesized digital designs fail. After a discussion of what causes both combinational and sequential logic to fail, the importance timing models in the synthesis process as well as concerns surrounding design time and portability of sub-threshold standard cell redesign are discussed.

4.2.1 Combinational Logic Failure

One of the primary reasons for poor sub-threshold operation of combinational logic cells is because their use was never intended for a large range of VDD operation. The optimal PMOS/NMOS width ratio for a super-threshold cell is different than the optimal ratio for a sub-threshold cell. Research in [64] found that in general, smaller ratios of around 1.2 are more optimal for sub-threshold operation. Cells with a large ratio of 3 are often not suited for sub-threshold operation, as rise times and fall times can be drastically different. Furthermore, these ratios can change from cell to cell as driving strength is changed to guarantee optimal performance in the super-threshold region. Unfortunately, this optimal sizing is heavily dependent on process-specific characteristics such as hole mobility, silicon straining, and varying threshold voltage differences due to different dopant implants which make operating characteristics different in sub-threshold.

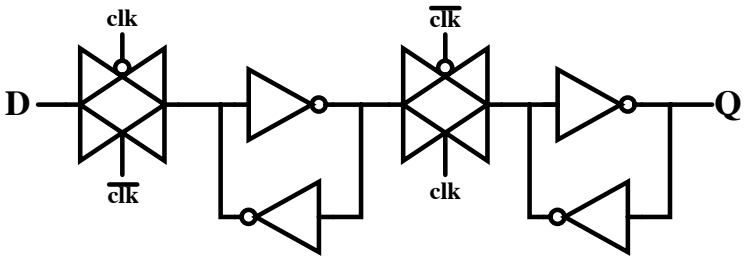
Moreover, there also exists the probability that no appropriate output may be

generated for a given input, similar to stuck-at faults. For example, large stacks of NMOS transistors can provide so much leakage current that a logic 1 (300mV) in sub-threshold will only output VDD/2 (150mV), which may not be enough to drive the next cell correctly [64], [59].

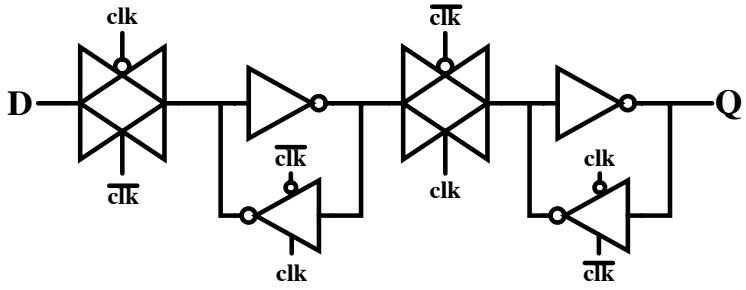
4.2.2 Sequential Logic Failure

Flip-flops exhibit their own set of complications. For example, most standard cell libraries are optimized for speed and area, by using a standard ratio-feedback flip-flop design consisting of fewer transistors [58]. This ratio flip-flop design results in possible failure when operated in the deep sub-threshold voltage regime because the driving strength of the input may not be strong enough to overcome the inverter feedback. For example, in Fig. 4.1 flip-flop (a) exhibits ratio feedback and therefore does not function properly across all process corners at its minimum energy voltage. However, flipflop (b) operates correctly because it utilizes ratio feedback in the form of clocked tri-state inverters.

In order for a synthesized design that utilizes ratio feedback flip-flops to operate in sub-threshold, all flip-flops need to be replaced by custom-designed flip-flops with clocked/enabled feedback. This process can be expensive and time consuming for a designer. Therefore, a simple solution to this is to utilize synthesized flip-flops to replace these poor-performing flip-flops in an automated manner. A simple Verilog netlist of inverters and tri-state inverters can be added to the design to insure proper flip-flop operation in sub-threshold. The design trade-offs for this



(a)



(b)

Figure 4.1: Flip-flop (a) contains ratio feedback and does not function properly at low sub-threshold voltages whereas flip-flop (b) contains tri-state feedback and does function properly at low sub-threshold voltages.

methodology of flip-flop replacement have not been explored to date.

4.2.3 Timing Model Inaccuracy

Another inherent problem with using standard cell libraries in the sub-threshold region (and probably the most critical) is uncertainty in the timing models. Each library has a variety of timing models for its nominal supply voltages, but not for sub-threshold supply voltages. Typically, an HDL synthesizer/compiler uses these conventional timing models to optimize the synthesized design and fix the

hold violations. When particular standard cells are operated in the sub-threshold region, their respective timing characteristics may not scale universally with other cells. While most cells have been shown to exhibit scaled-delays proportional to the supply voltage, some poorly-designed cells may not scale well with the rest of the library. Because the HDL compiler has optimized the design based on predetermined timings at the nominal supply voltage (i.e. 1.2V), using a cell with disproportionate delay can unknowingly violate setup/hold times when supplied with a sub-threshold voltage. This leads to longer delays and therefore more energy consumption due to increased leakage time.

4.2.4 Design Time and Portability

With conventional design practices, improving a library's ability to operate in sub-threshold is a very complex process. Typically, the time and manpower needed to redesign or re-characterize a library is either not available or too expensive. Furthermore, even if significant time is spent to alter a particular library, the process must be repeated again for every process library that is used. Various researchers propose redesigning cells for sub-threshold operation using expensive and slow tools such as Cadence's SignalStorm or Prolific's ProGenesis software. Unfortunately, two major drawbacks exist to using this technique:

1. A simple automated redesign does not optimize for area, speed, or device placement; it simply places transistors and wires them together with respect to a netlist. As a result, these redesigned cells will not be designed to the

same caliber as those within commercial standard cell libraries.

2. Designing cells strictly for sub-threshold operation worsens their operating characteristics in super-threshold (nominal VDD). This can be a hindrance when using dynamic voltage and frequency scaling [15] where the library needs to operate at both super- and sub-threshold voltages.

4.3 Proposed Near/Sub-Threshold Characterization Method

A three-step strategy is proposed for determining and eliminating standard cells that operate poorly in the sub-threshold region. This test measures the transient delay of each cell output at two different voltages, checking for consistency in the input-to-output proportional delay ratio between super-threshold versus near/sub-threshold supply voltages. If a cell's sub-threshold delay deviates too far from the nominal supply voltage delay, the synthesized cell will likely ruin a sub-circuit's ability to guarantee setup/hold time requirements in sub-threshold operation, as well as exhibit more energy consumption. Fig. 4.2 illustrates the proposed three-step process: parsing, testing, and removal. These steps are explained in the following sub-sections.

4.3.1 Parse Standard Cell Liberty File

Perl routines were written to generate and parse Spice simulations of every possible input condition for every logic cell, in any standard cell library operating at mul-

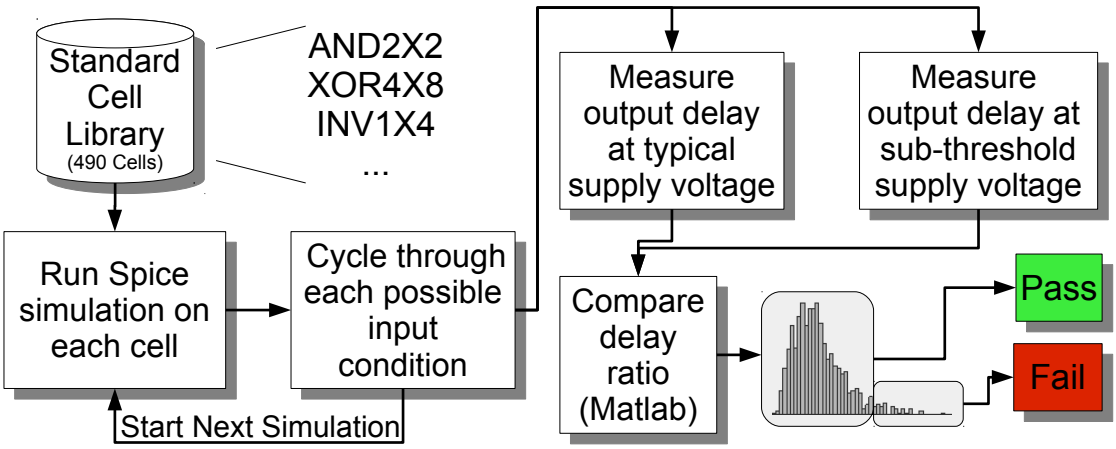


Figure 4.2: Functional diagram of the method used to determine poor cells and their removal from the sub-threshold standard cell library.

tiple supply voltages. The Perl script was written to parse a Synopsys standard cell Liberty file, an industry standard format [65].

All the necessary information for the logic gates is extracted, and a binary truth-table of input/output pairs is generated for each Spice input stimulus. This table is then analyzed to determine how to best generate both rising and falling edges at the output.

4.3.2 Input-to-Output Delay Variation Test

Once the inputs have been determined for each cell, each input condition is then exercised twice: once at nominal supply and once at sub-threshold supply. The input-to-output delay is measured for each supply and the difference is saved for

comparison. This process takes about 15 seconds per cell on a single core machine. The two resultant data sets are compared and the delay ratio of typical to sub-threshold supply voltages is calculated. Each data point is stored again in a table for later analysis in Matlab.

During the simulations, each input and output is connected to/driven by a NAND2 gate to guarantee uniformity in driving power. The NAND2 cell itself doesn't affect the simulations timing as the data is all normalized and relative for every cell.

4.3.3 Analysis and Cell Removal Decisions

Once every spice simulation is generated and tabulated, the data is analyzed and cell removal decisions are made. If the proportional delay of a cell is outside of the standard deviation of all the other cells, it is marked accordingly and can be removed. Also, a cell that is unable to generate a valid transition will be marked for removal.

Figure 4.3 displays the population of cells with a specific delay ratio in a 90nm-CMOS standard cell library across nine different near/sub-threshold supply voltages. The lower right corner of the figure shows several outlier cells (marked darkly), which are annotated because they did not generate an output voltage transition in sub-threshold. In the case of this particular library, the standard deviation of the delay variation (σ) was approximately 60x for VDD=300mV. Therefore, if 1σ delay culling is used, cells between 60x and 1000x will be removed.

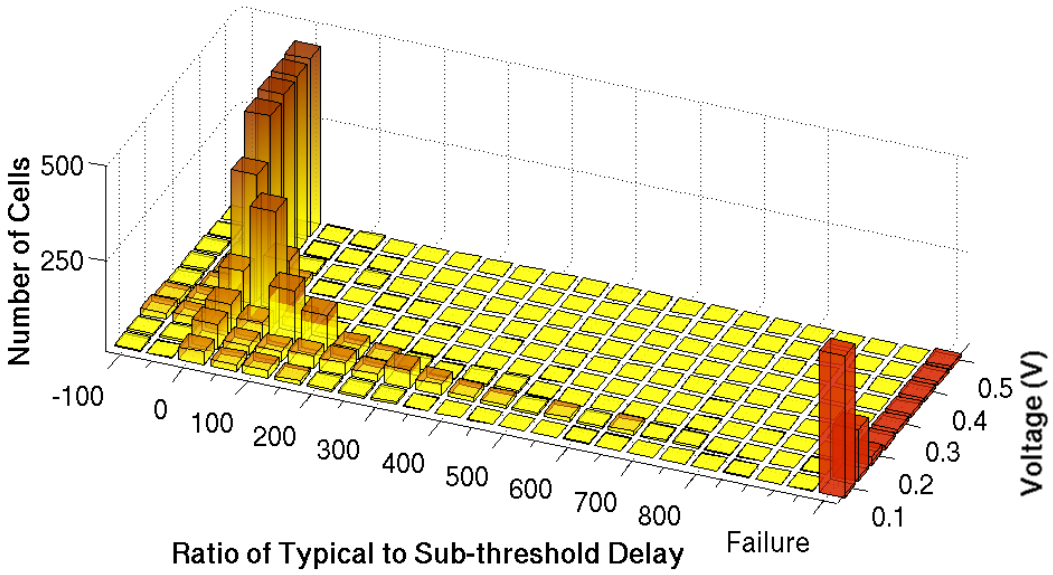


Figure 4.3: Histogram of input-to-output delay variation from near/sub-threshold to nominal voltage in a low-power 90nm cell library.

If 2σ culling is used, far less cells are removed.

Figure 4.4 illustrates how cells are chosen to be removed based on their relative delay radio from super- to near/sub-threshold normalized to the NAND2X1 cell of the library. In this selection of cells it is clear that the CLKINVX12 cell has the worst performance relative to the rest of the cells. Depending on how tight the timing constraint is picked to be (1σ to 2σ in this example) different cells can be removed. In this case, when 1σ culling is used three cells are removed (CLKINV12X, NAND4X4, XOR2X16) whereas when 2σ culling is used only one cell is removed (CLKINVX12).

Cells removed based on timing show a strong correlation to poor energy per-

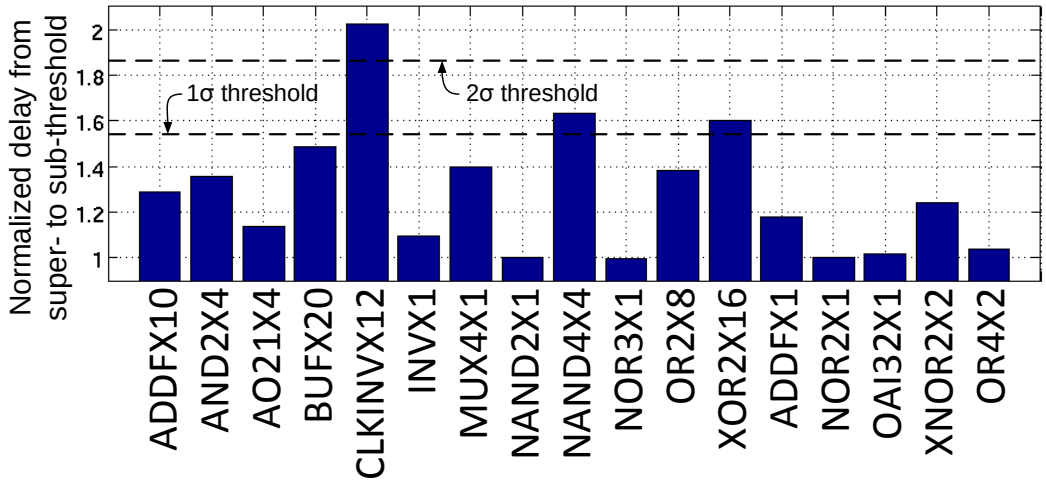


Figure 4.4: Bar plot of input-to-output delay simulations from near/sub-threshold to nominal voltage in a low-power 90nm cell library.

formance. In the case of an inverter, transistor sizing can play an important role in energy. The sub-threshold leakage equation [66] shows a clear dependence on transistor width, length, and threshold voltage V_{th}.

This means that, in the case of a poorly sized inverter, one transistor may leak more than the other. This results in a slower switching time and higher leakage current for one transition case (either low-high, or high-low). These mismatched transition speeds, when averaged together can result in a larger-than-average mean delay.

4.4 Results

Because of its versatility, this design automation methodology can be used to characterize any standard cell library. This methodology was tested on a 1.2V, 90nm-CMOS LVT standard cell library. For the purposes of this research, two different sigma values for the delay were tested within this culling process. First, cells that were outside of a 2σ delay bound were removed. Second, cells that were outside of a 1σ bound away from the mean delay were removed. In order to verify the procedure on synthesizable RTL, both a 16-bit Multiply-Add (MADD) and 32-bit Floating-Point Add (FP-ADD) HDL were synthesized. MADD and FP-ADD were chosen because of their high level of unique complexity. Three 90nm-CMOS netlists were created for each design: conventional non-culled (490 total cells), 2σ culled (35 cells removed), and 1σ culled (216 cells removed). The only differences between the three synthesized designs were simple statements in the compiler stating not to use culled cells in the synthesis process.

This methodology was also applied to a 65nm CMOS technology, with similar results. For example, the 2σ test with the 65nm library (890 total cells) culled 21 cells, 14 of which were inverters of various sizes. The characteristics of the removed cells are described as follows. In the rest of this chapter the area, delay and power improvements by using the proposed culling method are discussed.

4.4.1 Characteristics of Removed Cells

Due to the different standard cell structures and topologies there is no set of characteristics that all failed cells exhibit. However, as was explained in Section 4.2 the reasons can be bad PMOS/NMOS width ratios, large transistor stacking, and cells designed for super-threshold speed (which often have large widths). For example, the PMOS/NMOS ratio of the XOR2X16 cell that was tested was 2.8; by lowering it to 1.2 the cell can pass the 1σ delay test in section IV. Furthermore, this cell exhibits large transistor stacking resulting in unsymmetrical leakage paths in sub-threshold. By changing the topology of the cell performance may be increased further.

By removing cells with poor timing performance, the total energy of a design can be improved. Cells that exhibit poor timing are much more likely to have stronger asymmetric leakage currents. A typical sub-threshold design will operate at its optimal energy point where both dynamic and leakage energy are approximately equal. By lowering the unwanted leakage current paths, The design will be able to operate at an even lower optimal energy point as shown later.

4.4.2 Area Improvement

One of the major concerns when culling cells from a library is that in order to maintain the same speed and functionality, area may need to increase due to the loss of the culled, high driving strength standard cells or compact cells such as full adders. Most researchers in the field believe that the extra number of lower drive-

strength cells that replace the removed larger drive-strength cells will consume more area. However, from experimental results, this area increase is not observed for different synthesized test cases. Area decreased in all designs when cells were removed from the synthesizer. This area decrease also exhibits a positive effect on energy consumption. Table 4.1 compares the area of the six designs. For example the area of the MADD implemented using 1σ threshold culling is$10649\mu m^2$ vs. the MADD implemented without the culling technique which consumes $10988\mu m^2$.

Table 4.1: Area Comparison of Culled Designs with Unculled Designs

Netlist	Area [μm^2]	Total Cells in design	Cells Removed from Library
MADD normal	10988.07	2251	0
MADD culled 2σ	10785.56	2222	34
MADD culled 1σ	10649.46	2425	216
FP-ADD normal	5434.22	1026	0
FP-ADD culled 2σ	5391.96	1059	34
FP-ADD culled 1σ	4786.08	1346	216

4.4.3 Delay Improvement

For the MADD and FP-ADD examples, the culled designs showed better timing delay characteristics compared with unculled designs. Because operation in the sub-threshold regime increases delay sensitivity to process variation, 1000-point Monte Carlo tests were run to measure the worst-case operand delay for all six netlists shown in Section 4.1. Figure 4.5 shows an overlay of two histograms: the normal case and the 1σ culled case of an FP-ADD. Here, the mean delay decreases for the culled case, with standard deviation improving and fewer outliers present.

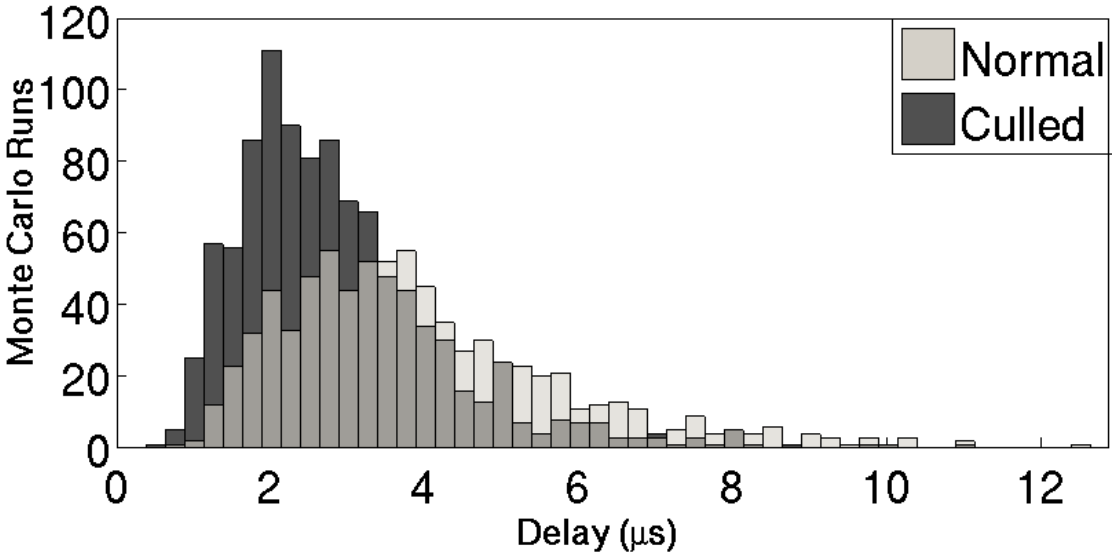


Figure 4.5: Histograms of FP-ADD delay with and without cells removed from synthesizer using 1σ culling method running at Vdd=400mV.

All culled designs (MADD and FP-ADD) show improved standard deviation of delay by around 25%. The FP-ADD design showed improved means of approximately 28%. The MADD designs did not exhibit mean improvements nearly as large, with one netlist showing a negative improvement. However, the worst case delay with that netlist was improved by 28% and the culled design exhibited a 63% reduction in outliers over the unculled design. Figure 4.6 shows two sets of box plots. Each set displays the mean and standard deviation of each design. The mean and standard deviation for each simulation are displayed above each bar.

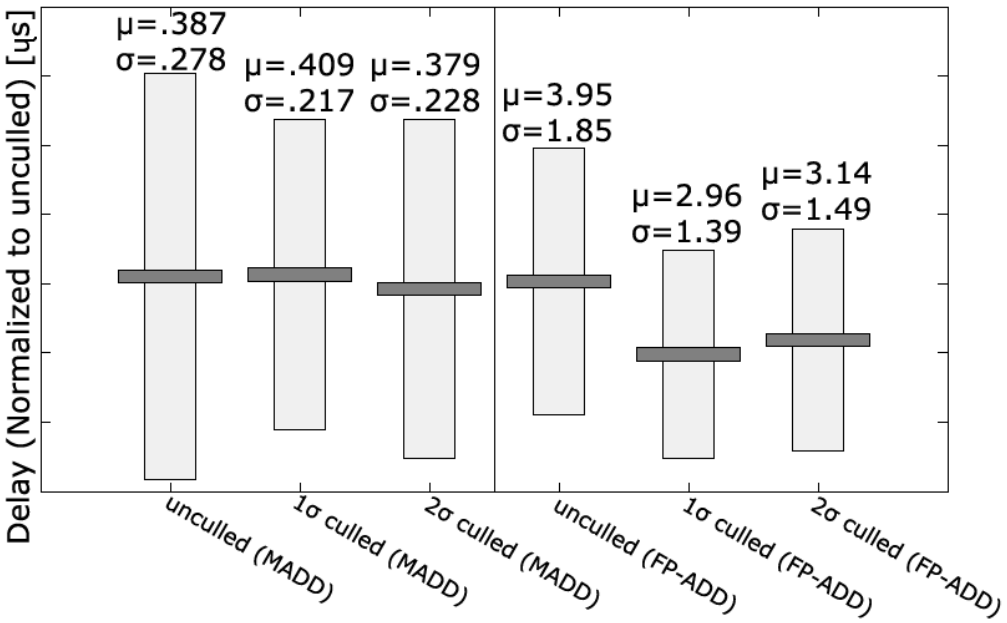


Figure 4.6: Box-plots of delay without and with cells removed from the synthesizer for both designs.

4.4.4 Energy Improvement

Because cells with larger delay variation have larger/longer unwanted leakage currents, removing offending cells lowers the overall integrated leakage energy due to the shorter clock cycle times. One major benefit of this culling methodology is the improvement of both unwanted leakage energy and overall energy/computation. All culled designs had lower energy than their unculled counterpart. The standard deviation improvement for energy/computation of both 1σ, 2σ culled MADD netlists were greater than 71% over the conventional, unculled case. Figure 4.7 compares the total energy/computation of all six designs which is directly related to a reduction of cells with unwanted leakage paths.

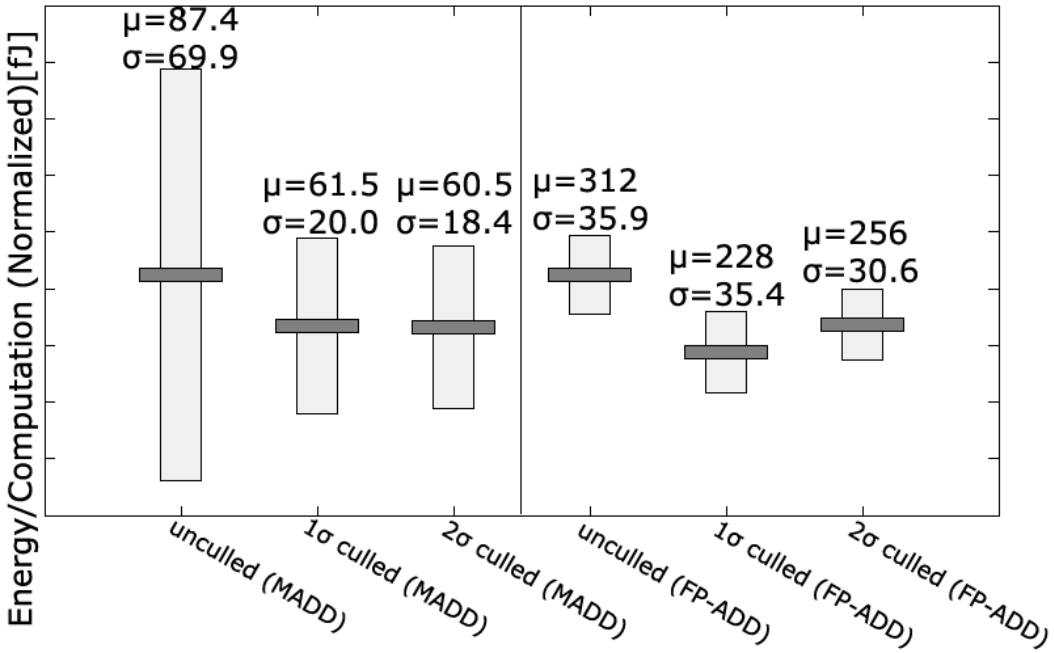


Figure 4.7: Box-plots of energy/computation without and with cells removed from the synthesizer.

Table 4.2: Leakage energy comparison of culled designs with unculled designs.

Netlist	Mean [fJ] (Improvement)	Standard Deviation (Improvement)
MADD normal	50.7	37.9
MADD culled 2σ	46.1 (10%)	23.0 (65%)
MADD culled 1σ	47.7 (6%)	24.3 (56%)
FP-ADD normal	94.3	21.2
FP-ADD culled 2σ	68.5 (36%)	17.8 (19%)
FP-ADD culled 1σ	56.0 (68%)	15.4 (38%)

In order to illustrate the potential energy improvement from using this method, an unculled FP-ADD netlist was simulated at its typical super-threshold voltage of 1.2V. A total energy/computation of 3.66 pJ was simulated. Comparing this to the unculled design in sub-threshold, energy/computation of 312 fJ was simulated.

This yields an improvement of about 10x as expected with sub-threshold operation. Comparing the unculled super-threshold energy to the culled energy of 228fJ yields an energy improvement of over 15x. It is clear that this methodology is extremely useful for improving sub-threshold energy of synthesized circuits.

4.5 Conclusions

This work describes a reliable, fast, and cost effective way to improve the operation of any standard-cell synthesized design in sub-threshold. The method described in this work is completely scalable as logic cells only need to be characterized once before the synthesis process which enables designs utilizing anywhere from hundreds of thousands to billions of transistors. This method improves timing, area, and energy dissipated per computation. Using this method allows for better DVFS response across multiple voltages within the super- and sub-threshold regions. Finally, by removing many outliers from a design, the worst-case performance can be improved resulting in better yield across many chips.

Future work includes expanding this methodology to sequential logic and verifying the operation of a complete synthesized microprocessor/ASIC, and verifying these simulation results with an experimentally measured test-chip prototype. Furthermore, this work can easily be adapted to test cells under a different set of constraints. For example, instead of using the mean Monte Carlo delay for each cell, the cells could be culled based on a worst case delay under Monte Carlo or a particularly bad process corner.

Chapter 5: Energy Constrained Encryption: A Case Study in Near-Threshold Circuit Design

5.1 Introduction

Encryption is becoming an important part of every-day data transfer. As applications are pushed to ubiquity and require tremendously low energy consumption, encryption will need to consume an ever-smaller amount. Conventional algorithms [67–69] requiring on the order of hundreds of μW to several mW will not be able to meet the energy and power budgets of future low-power systems.

For example, a typical Gen2 Radio Frequency Identification (RFID) tag requires its power consumption to be on the order of micro-watts. By lowering the power consumed by the electronics in the tag, less power is required to use it, resulting in a longer interrogation distance. There is also a fundamental power consumption limit where the tag will cease to function which is dependent on many factors such as tag design, carrier frequency, and antenna design.

The Hummingbird ultra-lightweight cryptography scheme was specifically designed for low-power applications and has the potential for very low power and low energy operation [70]. Being a combination of both block and stream ciphers,

Hummingbird can provide 256-bit security while maintaining a low overhead. Because of this, traditional RFID tags can be retrofitted with this encryption with little overhead for the additional security they provide [71]. However, as the typical Hummingbird implementation does remain within the power budget of a Gen2 tag, any additional circuitry (such as a sensor or additional computation/processing of data) will likely put the tag above its power budget. This added complexity then requires the encryption scheme to consume even less power than has conventionally been reported.

This chapter presents four different low-power encryption designs in order to push the limits of the RFID/internet-of-things application space. The designs are both architectural-level and circuits-level modifications of the state-of-the-art Hummingbird design [71]. Section 5.2 describes some of the ways the near-threshold challenges described in Chapter 2 are overcome. Section 5.3 contains a case study of the four different implementations of the Hummingbird algorithm highlighting the trade-offs required for each design. Before concluding, Section 5.4 presents the results from the case study.

5.2 Potential Solutions to Near-Threshold Challenges

There have been many solutions proposed to deal with the variation-rich environment of near/sub-threshold operation. This section discusses some of them as they pertain to small, low-energy systems.

5.2.1 Check and Adapt

One class of methods proposed to combat the highly variable conditions of near-threshold operation are known as adaptive scaling methods [72]. These methods usually combine some type of error detector such as Replica Circuits [73] or timing error detection circuits [22] with some type of prediction logic. If it has determined that errors are occurring at too frequent, different circuit parameters such as V_{dd}, core frequency, body biasing [72], and clock skew are scaled accordingly. Fig. 5.1 illustrates this feedback process on the systems level.

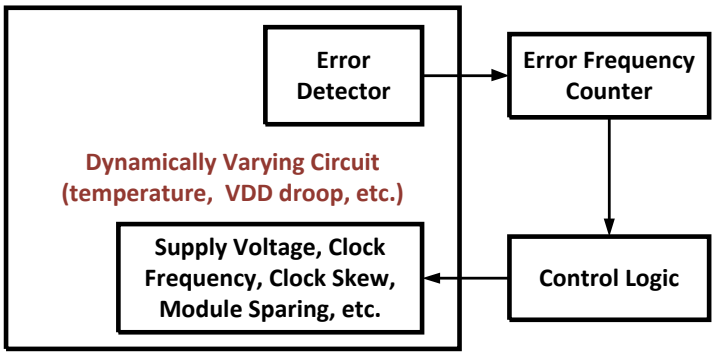


Figure 5.1: A systems-level diagram illustrating the feedback process of dynamically adapting circuit performance in the presence of dynamic variations.

These methods have been shown to improve near-threshold operation in large systems [73,74]. However, in small, energy constrained systems this type of solution will most likely have too large of an overhead to be practical.

5.2.2 Large Guard-Banding

All digital circuits with any timing constraints require a guard band of some sort. Engineers typically slow down the clock by some percent to guarantee timing in the presence of voltage droops, temperature changes, aging, and other variable factors.

The problem in near-threshold however, this that the guard-bands get much larger as voltage is lowered due to the exacerbated standard deviation that the variations exhibit. There could be an 80% increase or more in a guard band when going from super- to near-threshold voltages, making simple guard-bands non-ideal.

5.2.3 Architectural or Circuit Redesign

Instead of putting extra circuitry around an already working super-threshold circuit or adding a large guard-band to confirm near-threshold operation, a more ideal solution is to redesign either the architecture, circuits, or both for the specific purpose of operating at near/sub-threshold.

In *architectural redesign* an architecture already working in super-threshold can be redesigned to operate better in near-threshold. Some examples of this could be: designing architectures that intrinsically perform better in a variation-rich environment, or architectures that can easily adapt to variations on the fly.

In *circuit redesign* either custom circuits, or improved versions of already existing circuits are designed for near/sub-threshold operation. One example of full

custom near/sub-threshold circuits would be asynchronous circuits [75] (although these methods often require an architectural redesign as well). An example of improving a circuit design would be redesigning a flip-flop [76] or digital standard cell [74] for improved near/sub-threshold operation.

The obvious downside to redesign is the time spent in both the invention and design process as in near-threshold design it is often not straightforward how to improve a design.

5.2.4 Near-Threshold-Aware Synthesis Techniques

Another method that has gained popularity over the past several years is near-threshold-aware synthesis [77,78]. These methods generally fall into two categories:

1. Re-characterize a digital standard cell library and generate new timing information for the synthesizer to use in determining cell selection and placement [78]. This method is similar to simply re-characterizing a standard cell library with the advantage of adding variation information to the timing via Monte Carlo codes.

2. Determine which cells in a library are worthy of near-threshold operation and remove unworthy cells from the synthesis process [77]. On top of this, this particular method also correlates the timing of super-threshold operation to near-threshold operation allowing the synthesis tool to synthesize the circuit at super-threshold but be used at near-threshold with predictable timing. This method is presented in Chapter 4.

The two methods are similar in that their overall goal is to essentially reduce the amount of variations the final circuit will experience based on simulating with Monte Carlo codes. The first method is likely to be more comprehensive at the impact of the much longer CPU time required to regenerate new timing libraries. The second method's strength is in its ability to provide a much similar function at a much lower cost. However, both methods have the inherent drawback of requiring large systems in order to have more cells to tune. If the system is too simple there is likely only one circuit solution and thus no optimization can be done.

5.3 Case-Study

This section presents designs aimed to understand the abilities and limitations of the proposed method in Chapter 4. The designs are different implementations of the low-power Hummingbird algorithm and are presented in the form of a case study.

5.3.1 Hummingbird Encryption Scheme

The Hummingbird algorithm (Figure 5.2) consists of four 16-bit block ciphers EK_1, EK_2, EK_3, and EK_4, four 16-bit internal state registers $RS1$, $RS2$, $RS3$, and $RS4$, and a 16-stage LFSR. For each encryption request the majority of the computation is taken up by each EK module requiring 4 cycles.

The EK modules themselves consist of multiplexing logic that cycles through the 256-bit key (k_n), 64-bits at a time. The 64-bit sub-keys then go though more transformations including S-Boxes (S_1, S_2, S_3 and S_4) which act as lookup tables. During the fourth cycle of the EK module two S-Boxes are used as can be seen in the right half of Figure 5.2. This makes the standard encryption process take 16 cycles total to complete an encryption (from input to output) after a 69 cycle initialization [71].

The sequential structure of the Hummingbird algorithm lends itself as a good case study for efficient low-power digital design as many encryption algorithms are similar in nature. Furthermore, this particular algorithm allows for many different optimizations to be made which will be the primary focus of this paper.

5.3.2 Tested Solutions

Based on the application space described in Section 5.1 three different designs are studied in their ability to reduce the variation of the Hummingbird algorithm operating in near-threshold. Two of the designs (1 and 2) are architectural/circuit redesign solutions and the third (3) uses the near-threshold-aware synthesis technique presented in chapter 4 to reduce variations. Each design is compared to the original Hummingbird design operating in near/sub-threshold. These solutions were tested because they are cheap (in both design time and resources required), low-overhead, and easily adapted/ported to other applications and process nodes.

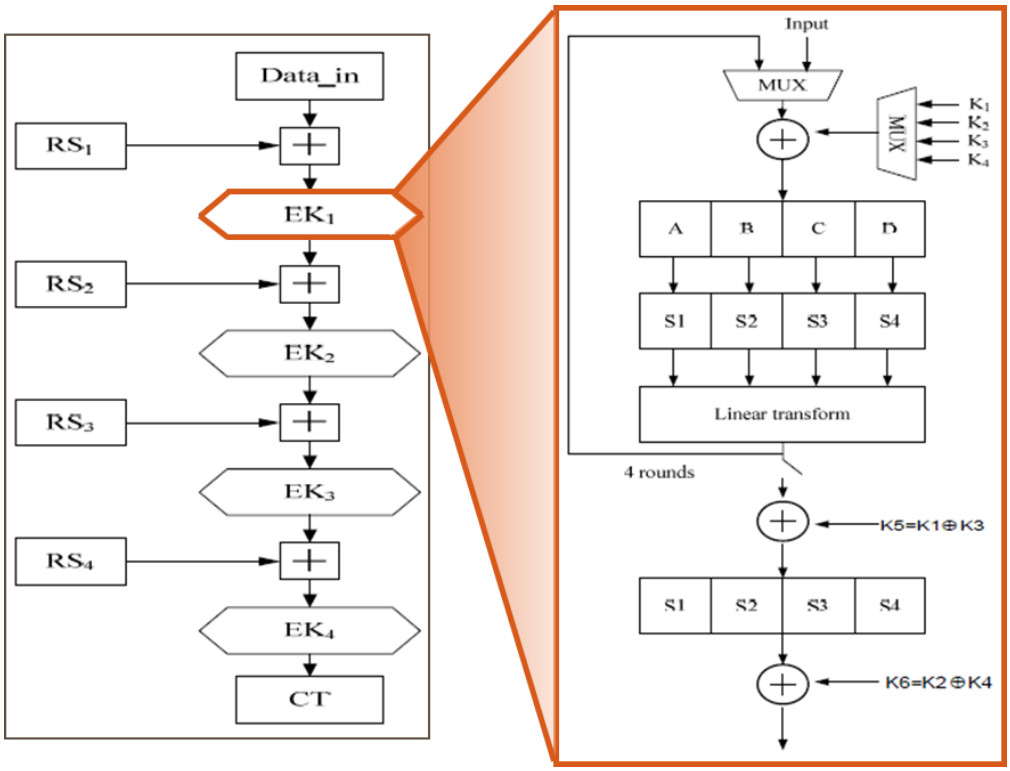


Figure 5.2: The Hummingbird cryptographic algorithm.

5.3.2.1 Test 0: Hummingbird with Reused EK Module

The first solution tested is the original hummingbird architecture [71]. In this architecture, in order to maintain a 16-cycle latency for each encryption the EK module can be reused. Figure 5.3 shows a more in-depth view of how the architecture is implemented. This version of the algorithm basically consists of multiplexers redirecting signals back through the reused EK module.

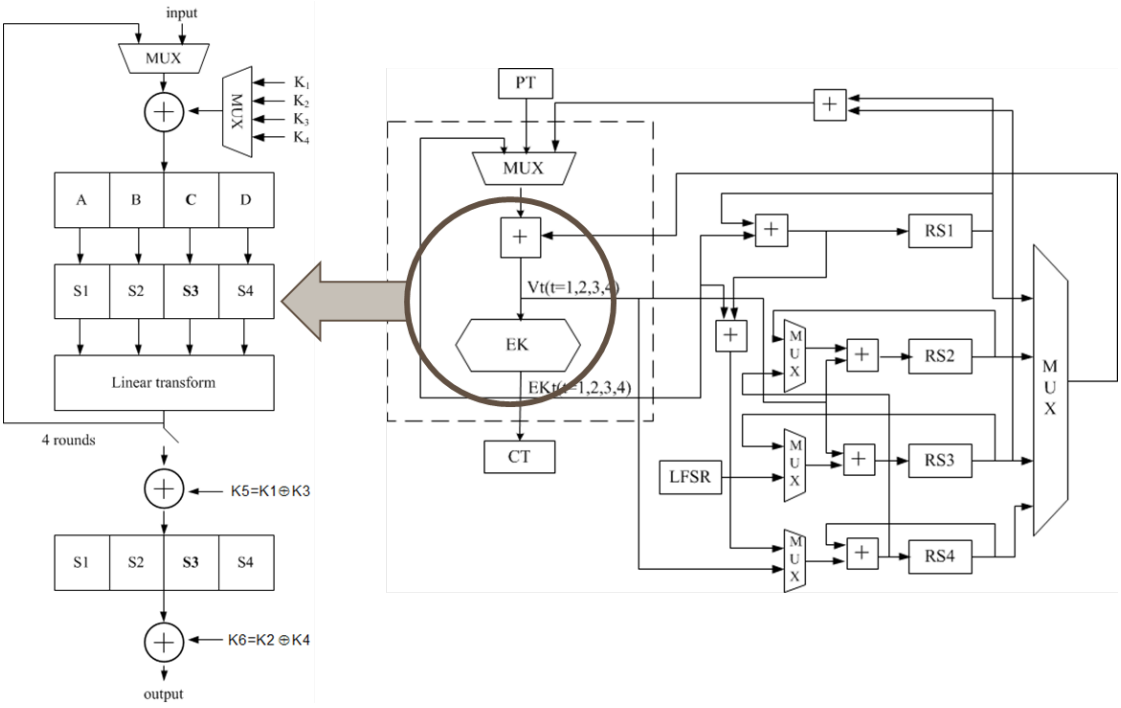


Figure 5.3: Original implementation of the Hummingbird algorithm.

5.3.2.2 Test 1: Area-Optimized Hummingbird

This version of the architecture was specifically designed to consume as little area as possible. This is done not only by the EK reuse as done in test 0 but by also reusing the S-Box within that EK module (Figure 5.4). This requires additional cycles to complete the encryption at the advantage of requiring less area and less power as there are fewer transistors used in the implementation.

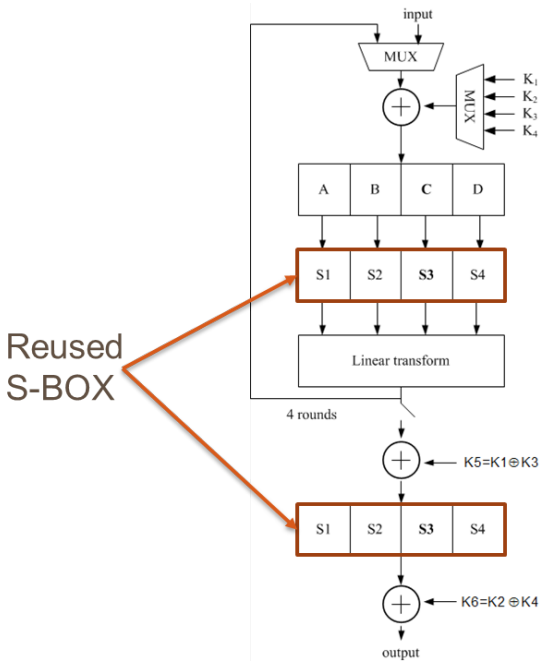


Figure 5.4: Area-optimized implementation of the Hummingbird algorithm.

5.3.2.3 Test 2: Loop-Unrolled Hummingbird

The final architectural redesign test takes the hummingbird architecture and unrolls all the looping done in tests 0 and 1 (Figure 5.5). Namely the EK module and S-Box within are no longer reused making the critical path much longer. This longer critical path in itself is aimed at reducing variation as longer logic chains amortize the variation of each element throughout the chain. The longer the chain, the less variation should be seen [79]. There is an obvious trade-off here in that this design may execute faster and require less power to operate.

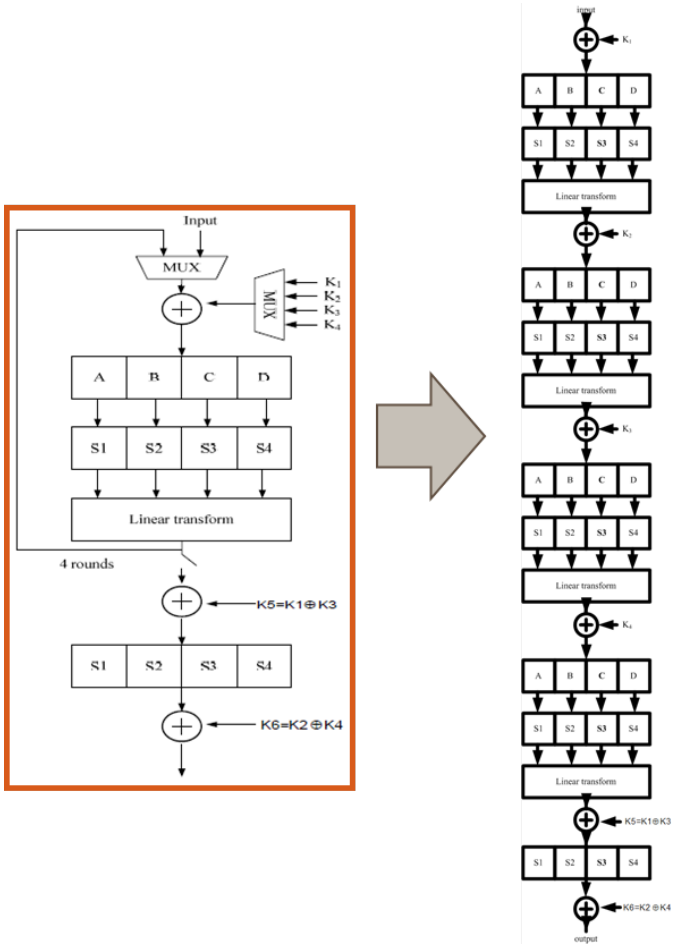


Figure 5.5: Loop-unrolled implementation of the Hummingbird algorithm.

5.3.2.4 Test 3: Test 0 with Synthesis Improvement

The final test focuses on the Near-Threshold-Aware Synthesis Technique presented in Chapter 4. In this test the same Hummingbird implementation as used in test 0 is used but after the automated standard cell culling process has been implemented on the standard cell library to be used to synthesize the final circuit.

5.4 Results

After designing and synthesizing each solution, 100-point Monte Carlo simulations were used to find the variability of the 12 worst-case paths for each design along with area, power, and energy.

Because the Hummingbird architecture was originally designed for use in an RFID tag the tests were carried out as follows. A Gen2 RFID tag operates at a frequency of $1.28MHz$ [71]. Therefore, for each implementation simulations were carried out to determine the lowest possible operating voltage in order to meet the $1.28MHz$ timing requirement given by the Gen2 requirements. At that voltage, all measurements, including variation were performed. The voltage found for each implementation can be seen in Table 5.1 on Page 84.

5.4.1 Area Impact

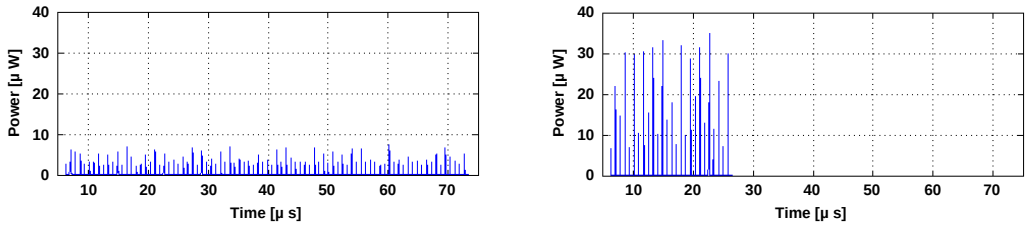
The area impact of each version is straightforward and is summarized in Table 5.1. The area-optimized version consumes about 5% less area than the reused EK version which is a very modest improvement given the addition of 19 clock cycles required to complete an encryption makes it 20% slower. The speed-optimized version is more than 20% larger than the reused EK version, but because it is now more than 3.5X faster this trade-off pays off from an area versus speed perspective. Finally the synthesis improvement version has almost no change in area consumption over the reused EK version. The area does grow marginally because the types of cells being culled are often minimum sized, requiring the synthesizer

to select some cells that are larger than what were picked before.

5.4.2 Energy and Power Impact

Energy was simulated for each test across the total cycles required to complete one encryption. Power was extracted from the simulation as an average throughout the same time. Both numbers are recorded in Table 5.1 for each test. In general, the power for each test scales with it's area and the energy scales with cycles. Furthermore, all tested solutions exhibit over 96% less power than any of the contrasted previous works also reported in the table. This is due primarily to the lowering of supply voltage while maintaining the performance requirements of a Gen2 RFID tag.

Figure 5.6 contrasts the power/energy trade-off for the shared EK and speed optimized designs. The energy required for the speed optimized version is dramatically reduced as the reduction in clock cycles overcomes the increased power required for the design. In a strictly power-constrained application the speed optimized version is not a better candidate over the shared EK version. However, it's important to note that future RFID systems will require extra computation in the form of sensor measurements and calculations which may reduce the amount of cycles allotted for the encryption which may make it a viable solution.



(a) Power simulation of Test 0

(b) Power simulation of Test 2

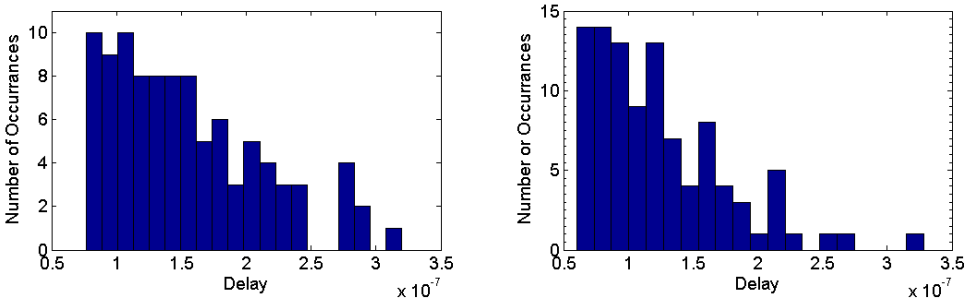
Figure 5.6: Power plot vs. time of test 0 (Shared EK) and test 2 (Speed Optimized).

5.4.3 Variation of Each Architecture

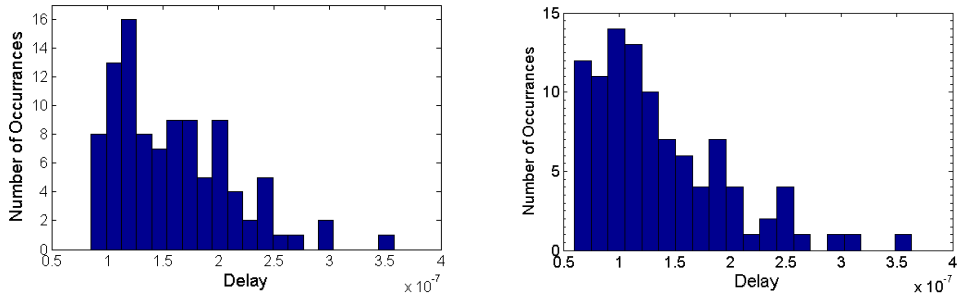
After compiling the Monte Carlo timing data for each test, the standard deviation and mean were calculated to be used in the common metric of $3\sigma/\mu$ which yields a quantitative measure of variability [79]. A qualitative measure of the variability of each test can be seen in Figure 5.7.

5.4.3.1 Variability of Test 0

The $3\sigma/\mu$ variability for this version is the best of the three non-speed optimized versions at 122% but marginally so that there is not a large difference in the measure. By looking at the histogram in Figure 5.7(a) the histogram looks much more heavy-tailed than any of the other plots. This yields the conclusion that the $3\sigma/\mu$ measure should not be the only metric used to determine variability and that this version may not give the best yield.



(a) Shared EK (b) Area-optimized



(c) Speed-optimized (d) Synthesis improvement

Figure 5.7: Variation Response of four test cases across Monte Carlo.

5.4.3.2 Variability of Test 1

The area-optimized version of the architecture has a 3% larger $3\sigma/\mu$ but as can be seen in Figure 5.7(b) it's histogram is a lighter tail than the previous test and most of the weight has been moved to the front of the distribution.

5.4.3.3 Variability of Test 2

The speed-optimized architecture has the best variability performance both in terms of $3\sigma/\mu$ and by a qualitative inspection of Figure 5.7(c). The long paths in this version of Hummingbird help amortize the variability across many gates resulting in a smaller deviation as the path is exercised both by different input vectors and different variations from chip-to-chip.

5.4.3.4 Variability of Test 3

For the most part, the variation of this test doesn't change much over the original test because its architecture is already so simple to begin with. As the synthesizer goes to change the cells after they are culled from the library it finds that there are very few replacement options that might result in an improvement. The $3\sigma/\mu$ for this test is the largest of all the tests and is 11% larger than the original version. However, this is mostly due to the outliers as the center of mass in Figure 5.7(d) are much more concentrated at the beginning of the histogram compared to the original (shared EK) version.

Table 5.1: Simulation results for each tested design

Architecture	Metric							
	Technology $[nm]$	Area $[\mu m^2]$	Cycles	Power $[\mu W]$	Energy $[nJ]$	VDD	Speed	$3\sigma/\mu$ [%]
Original Hummingbird [71]	130	14375	85	30.67	NR	1.2V	1.28 *MHz*	N/A
0: Shared EK	90	9904	85	0.29	13.1	370mV	1.28 *MHz*	122
1: Area Optimized	90	9447	104	0.28	19.3	390mV	1.28 *MHz*	125
2: Speed Optimized	90	12214	24	1.1	4.6	440mV	1.28 *MHz*	100
3: Synthesis Improvement	90	9941	85	0.3	13.6	380mV	1.28 *MHz*	133
256-bit AES [68]	180	43000	?	NR	19	1.2V	asynchronous	N/A
256-bit AES [67]	180	50377	500	224	NR	1.2V	10 *MHz*	N/A
1024-bit RSA [69]	130	270000	2	15000	NR	1.2V [a]	13.56 *MHz*	N/A

[a] Assumed based on technology node.

5.5 Conclusion

This chapter introduced four improved Hummingbird light-weight encryption scheme designs for use in future RFID systems. The designs aim to contrast different design trade-offs while operating in the near-threshold voltage regime. The designs reduce power from at least 96% to over 99% the most recently reported hummingbird I design. The designs add a 30% to 40% increase in the digital base-band area which is less than most encryption schemes. The shared EK design reduces hummingbird's power by over 99% by simply operating the encryption scheme at a near-threshold voltage of $370mV$. The area optimized design reduces hummingbird's area by 5% at the cost of 22% more cycles as well as exhibiting the lowest power of all designs. This design is only useful when area is the utmost concern. The speed optimized design reduces the power by 96% while reducing the number of cycles required of the algorithm by 72% and is a great candidate for cycle-constrained RFID systems.

On top of evaluating these trade-offs, the reliability (in the form of timing variability) of each near-threshold design was tested through Monte Carlo codes. While some designs did show better timing reliability performance there is no one design or design style that reliably reduces variation beyond a moderate level. While some methods [77, 78] have shown to reduce variation in large designs (such as the work in Chapter 4), a small design such as Hummingbird leaves little room for modifications resulting in less reliable improvements as seen by the synthesis improvement design reported in this article.

Chapter 6: Asynchronous Circuit Operation

6.1 Introduction

Asynchronous logic design has been studied for well over 60 years. Only recently has this paradigm actually been applied to practical applications [80–86]. Due to the complex nature of asynchronous circuit design, few techniques have been accepted by industry as cost-efficient alternatives to traditional synchronous designs. However, in this highly unpredictable regime of neat/sub-threshold, asynchronous circuits may become a viable option. At the very least they can give insight into how to design more reliable circuits in the future.

The following will explore in detail asynchronous techniques that aim to be competitive to traditional synchronous synthesized techniques used today. The primary methods explored are ways by which traditional synchronous systems can easily be modified for asynchronous operation. The most widely researched method for doing this is known as completion detection within an asynchronous micropipeline.

This chapter starts with an introduction to the fundamentals of asynchronous circuit design and a discussion of its challenges are presented. After a review of

the asynchronous micropipeline, a comprehensive analysis of existing completion detection techniques is presented. Finally, two new completion detection methods are presented and compared.

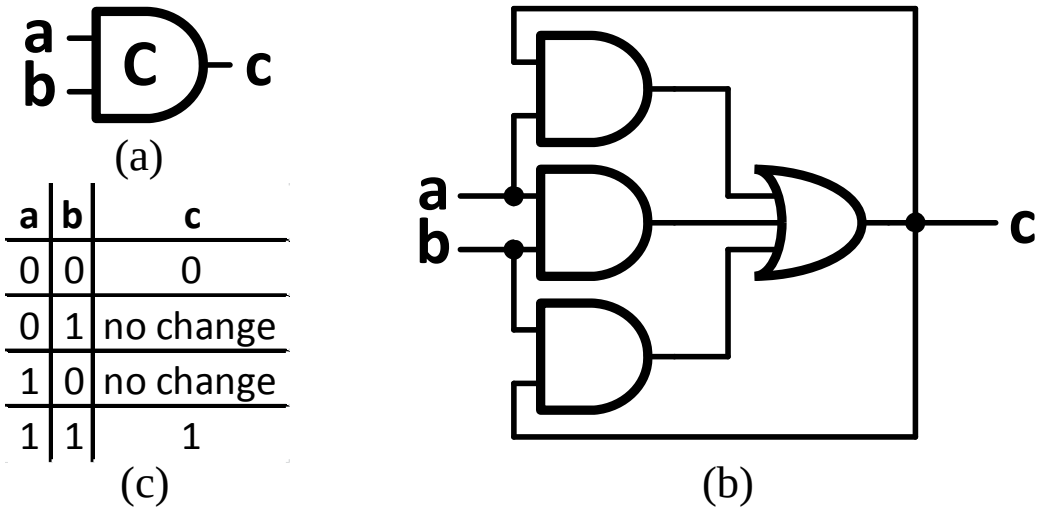
6.2 Traditional Asynchronous Techniques and Challenges

6.2.1 Asynchronous Logic Elements

One of the most important components of modern asynchronous systems is the *Muller C-Element* (also known as the join or rendezvous element) named after its creator David E. Muller (Figure 6.1a). When its inputs are identical, the output matches the inputs. Otherwise, the output doesn't change (Figure 6.1c).

This component is commonly used for handshaking control circuits and is most often the fundamental building block in delay-insensitive circuits [87]. The C-element is widely researched and there have been many implementations [88–91]. In [89], a synthesize-able version is described, which is shown in Figure 6.1b.

There are many other asynchronous components aside from the C-element [90, 91]. The *Toggle* component is often used in asynchronous counters and many forms of control and arbitration logic. It can also be used to convert between micropipeline handshake mechanisms. The toggle component symbol is shown in Figure 6.3b. The *Merge* component is functionally identical to the discrete logic XOR and is often used when more than one block needs to communicate to one output [92]. In Figure 6.3c it is also used to aid in the conversion in micropipeline



a	b	c
0	0	0
0	1	no change
1	0	no change
1	1	1

(c)

Figure 6.1: Asynchronous Muller-C Element: (a) schematic symbol, (b) Synthesize-able, hazard-free circuit schematic [89] (c) C-element truth table.

handshaking mechanisms.

6.2.2 Hazards

In synchronous systems, glitches (0-1-0 or 1-0-1 transitions) in outputs can be tolerated as long as they don't violate setup and hold times. However, many forms of asynchronous systems require glitch free operation to ensure datapath correctness. Asynchronous systems call these glitches *hazards* [93]. Some techniques aim to filter out hazards (typically with some type of inertial delay [94]). Others stop them from happening altogether with careful logic design. [89]. Figure 6.1b shows an example of a common hazard-free C-element logic gate. In this case, extra logic gates that don't alter the truth table are used to unsure hazard-free operation.

6.3 Asynchronous Micropipelines

The fundamental method for controlling asynchronous circuits is the micropipeline [88]. Pioneered by Ivan Sutherland in the late 1980's, the micropipeline (or μ-pipe) is essentially the delay-insensitive version of a synchronous pipeline. Instead of the control of the pipeline being based on a global clock signal, registers or latches between stages are triggered by events from within the stages themselves.

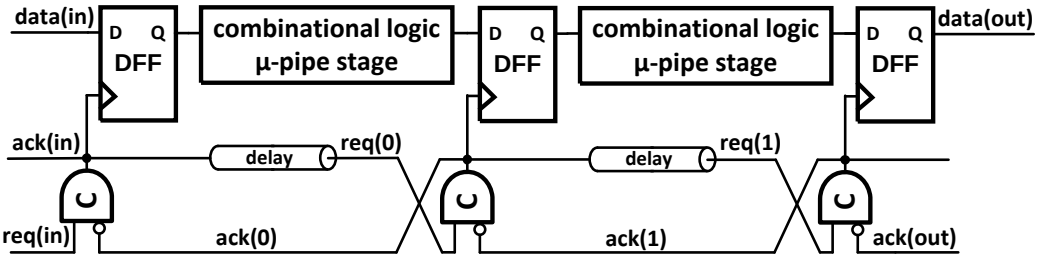


Figure 6.2: A fundamental μ-pipe with 2-phase bundled data control via C-elements

6.3.1 Micropipeline Configurations

The fundamental μ-pipe configuration is shown in Figure 6.2. In this case a two-phase bundled-data configuration is used to send *request* (req) and *acknowledge* (ack) commands between μ-pipe stages where each μ-pipe latch is controlled by a network of C-elements. Note that the C-elements need to be initialized to the correct state on startup. This can either be done with extra reset logic or a signal can be built into the C-element's structure.

6.3.2 Handshake Mechanisms

After Sutherland's pioneering work, many different μ-pipe control schemes have been designed to facilitate different signaling protocols and μ-pipe configurations [95]. They all rely on two primary signaling conventions illustrated in Figure 6.3a.

6.3.2.1 2-Phase Bundled Data

This convention is the easiest to realize with C-elements. It requires the least amount of logic and uses a small amount power because every transition (both rising and falling) is considered an event that latches data into the next μ-pipe stage.

6.3.2.2 4-Phase Bundled Data

With this convention, only rising edges constitute an event. This means that for each event the signal line needs to be charged and discharged. Although this consumes more power, it is much easier to produce these signals with most completion detection methods and is easier to control latches with.

6.3.2.3 Handshake Conversion

In order for 4-phase completion detection methods to communicate with 2-phase micropipelines, conversion circuits are needed. The asynchronous Toggle element

(easily realized for this application as a toggle flip-flop) can easily convert from 4-phase to 2-phase (Figure 6.3b). The simple pulse generator in Figure 6.3c can generate a 4-phase signal from a 2-phase signal.

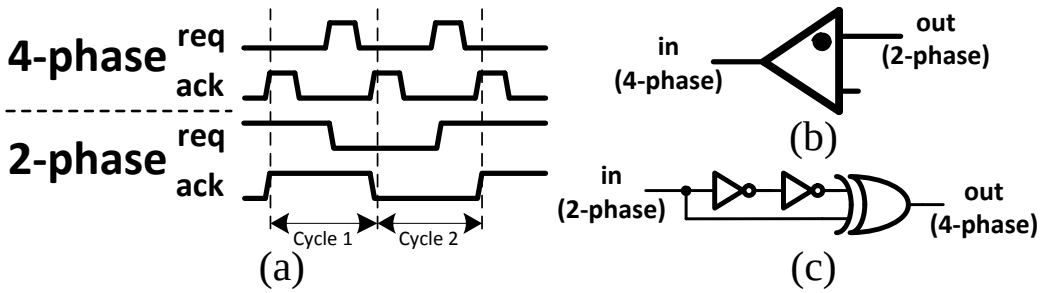


Figure 6.3: (a) 4-phase and 2-phase bundled data handshaking conventions, (b) 4-phase to 2-phase conversion with the Toggle element, (c) 2-phase to 4-phase conversion with the merge (XOR) element.

6.4 Asynchronous Completion Detection

One critical bottleneck to the widespread adoption of asynchronous circuits is completion detection. Modifying a synchronous pipeline to an asynchronous μ-pipe intrinsically does not improve performance of the intra-stage logic. In order to gain the speedups praised by asynchronous logic a scheme must exist to detect when the logic has completed so that the average-case delay can be exploited.

The following explores the existing methods used to adapt synchronous logic to asynchronous operation. Comparisons are drawn between these methods and their advantages and disadvantages are discussed.

6.4.1 Custom Circuits

Because of the complexity of asynchronous circuits there are a myriad of ways to implement the same logic with different types of circuitry. This has led circuit designers to come up with clever implementations to save area and power as well as increase the speed of this class of circuits. Some examples of custom circuit design are presented in previous research [75, 90, 96].

A popular form of custom logic is Differential Cascode Voltage Switch Logic (DCVSL) [96]. This works by pre-charging all logic gates to a "not done" state (all outputs: logic 1). Dual-rail signaling [97] is then used to pass signals between gates and a pull-down network is used to determine the value to be passed to the next gate. Another example of a full-custom technique is shown in [75]. In this paper it is shown that the implemented asynchronous circuits can operate reliably down to 150mV without the need for clocking adjustments.

This type of logic can be very time consuming to design and can result in an area overhead of over 40% compared to synchronous versions of the same logic [96]. Because every gate usually has to be charged and discharged for every computation, designers have to be clever in order to find speed-up opportunities by bypassing unneeded logic when possible in order to counter the larger power for these designs [75].

6.4.2 Null Convention Logic

Null Convention Logic (NCL), along with its many permutations: NCL-X [98], 2NCL, 3NCL [99], is a method by which delay-insensitive circuits can be synthesized via a dual-rail or similar encoding technique [97].

NCL circuits have a wide variety of uses and have been well researched [99–101]. In order to make them synthesize-able, a specific list of 27 NCL gates are defined that the NCL synthesizer understands [102]. Using a conventional language such as Verilog/VHDL the synthesizer can take a library of NCL gates and synthesize a complete asynchronous system [103].

It has been shown that the ability to synthesize NCL designs does decrease the area footprint over manual NCL designs by around 15% on average. However, using NCL versus standard synchronous synthesis can result in well over 100% more area after the synthesizer's optimizations [101].

6.4.3 Matched Delay Lines

The earliest, and most popular method of converting conventional synchronous circuits to work with asynchronous micropipelines is a technique known as *matched delay lines* (MDL). With this technique, a standard synchronous logic block is synthesized (multiplier, adder, etc.). The worst case path of that block is then replicated and placed in the *req* signal's path as shown in Figure 6.4. This matched delay must be slower than the functional block under all physical conditions and all data input cases (including the setup time of the next stage of the μ-pipe).

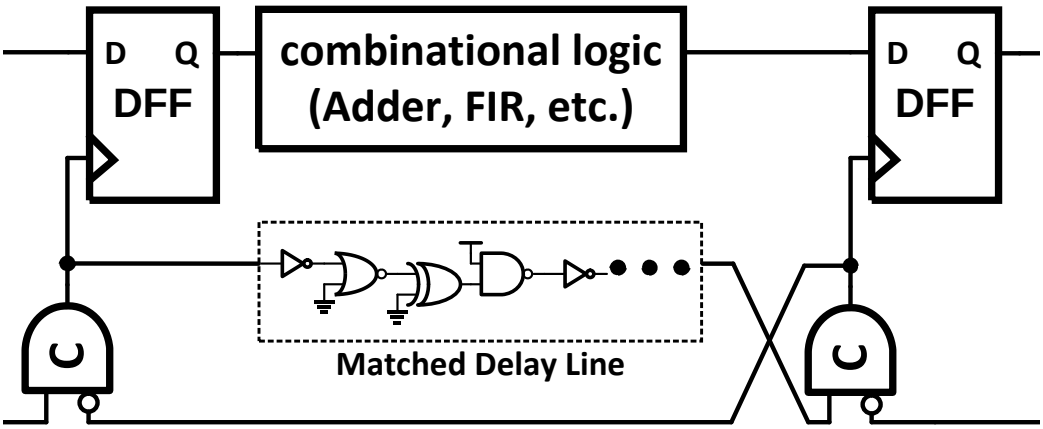


Figure 6.4: An asynchronous μ-pipe with completion detection by means of a *matched delay line*.

MDLs are one of the most widely used techniques in micropipelines to date [86, 95, 104–106].They are very easy to design/synthesize, require small area overhead compared to other asynchronous techniques, and work with any type of logic. However, the obvious downside is that there is no way to exploit the average-case performance. The μ-pipe can only run at the speed of it's slowest stage, just like a synchronous pipeline. The main advantage to using a MDL over a synchronous pipeline is the removal of the global clock network.

6.4.4 Speculative Completion

In [107], Steven M. Nowick proposed a speculative approach to improve the performance of MDLs. Instead of having only one worst-case delay line, two or more delay lines are used: one for worst-case timing and the remaining one(s) for faster speculative completion. In the case of [108], two extra delay lines are tuned faster

to facilitate early completion when a speculator determines a speedup opportunity is possible. Figure 6.5 shows the architecture of a speculative completion system. The key feature of this system is the *abort detection network*. The abort detection network makes sure the worst-case delay line is used when a speculation is unverified or is taking too long to compute. The one requirement of the abort detection network is that it becomes stable before the delay of each fast delay line. The networks are highly dependent on the logic they are speculating and can be difficult to design [108].

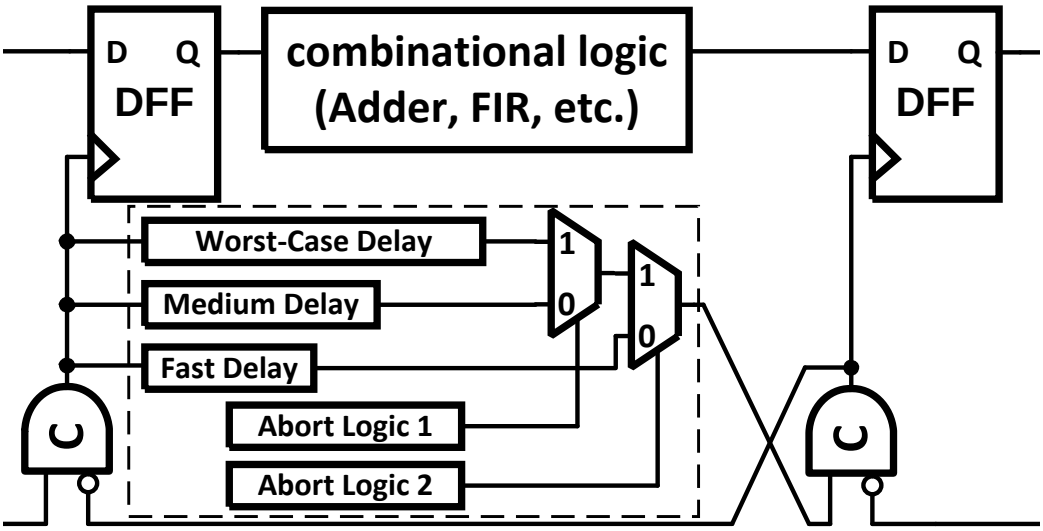


Figure 6.5: An asynchronous μ-pipe with completion detection by means of *speculative completion detection*.

Just like the simple MDL technique this version of completion detection is easy to synthesize using traditional techniques and works with 2-phase signaling. Unlike a simple MDL, this technique actually allows for a speedup to be achieved, providing an opportunity for average-case performance. The unfortunate downside

to speculative methods like this are that they are very architecture dependent. This method may work for a Brent-Kung adder with the proper modifications, but not a simple ripple-carry adder.

6.4.5 Current Sensing Completion Detection

Current Sensing Completion Detection (CSCD) [109–116] consists of an analog sensor that senses the current going through a group of combinational logic via a resistor or power gate transistor as shown in Figure 7.3. As a system starts to compute on new data, its current consumption increases. After the current consumption abates to a steady state, the computation can be considered completed and an asynchronous *req* signal is generated to send to the next μ-pipe stage. Multiple sensors can be combined in large systems to ease the requirements of sensitivity. This could be anywhere from a sensor on each μ-pipe stage [116] to a sensor on each custom-designed standard cell [112].

The advantages of this class of completion detection circuitry are: (1) a relatively low overhead compared to other methods such as NCL; and (2) the speed-up achieved, as there is finer speedup granularity as compared to speculative methods [115]. The complexity of the analog current sensor can discourage digital designers as they are difficult to combine with traditional digital synthesis. Additionally, if the current sensor restricts the current flow to the combinational logic the result will be a slower computation than one without the sensor.

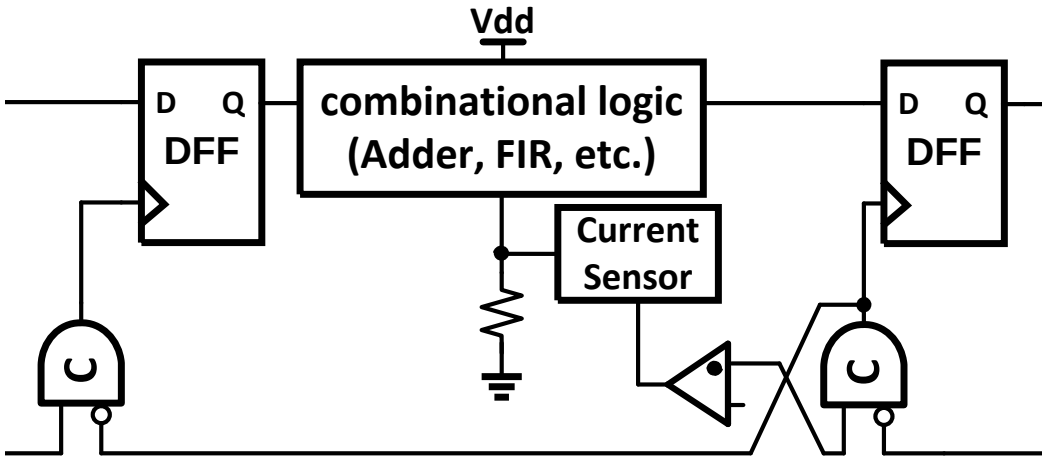


Figure 6.6: A standard combinational block modified for asynchronous operation by means of *current sensing completion detection*.

6.4.6 Sense-Inverter Based CSCD

In [117], a method is proposed that mitigates the concern that a supply voltage droop (as in CSCD) can have negative effects on logic speed if not designed carefully. *Sense-inverter based CSCD* is functionally similar to CSCD in the way that the current sensor is designed. However, in this method the current sensor is not placed in series with the logic's power supply.

Inverters are strategically placed at critical points within a synchronous circuit. In the case of a large adder, they are connected to the carry out pin of every full adder. These "sense-inverters" are then connected to a separate power supply that is in series with the current sensor (Figure 6.7). As the carry pins toggle, the inverters pass current through the current sensor. Completion is considered when the current consumption of the sense-inverters abates which is dictated by the cessation of switching activity on all carry signals.

The two clear advantages to this method are: (a) removing the current sensor from the datapath supply guarantees no speed penalty due to virtual power supply droop. However, it should be noted that the addition of this circuitry adds a small amount of capacitance to each node that may affect performance. (b) The constraints of the current sensor can be relaxed as more input dynamic range can be achieved because the droop can be larger on the sense-inverter's supply.

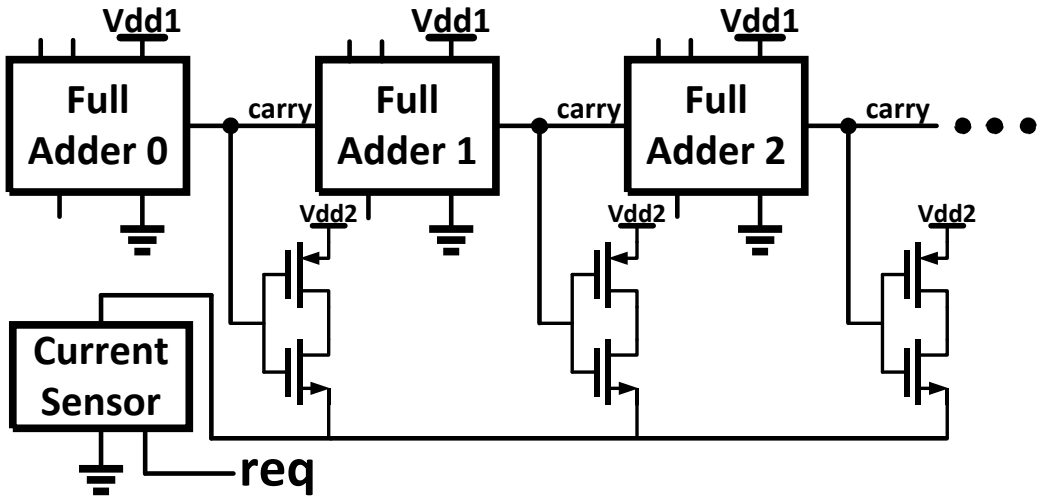


Figure 6.7: Basic configuration of *sense-inverter based current sensing completion detection* within an adder structure.

In order to make sure this method will work correctly, the sense-inverters need to be placed frequently enough within the datapath such that there will never be an accidental lapse in switching activity to the sense-inverters. This fact along with the more complicated design constraints and the coupling with an analog current sensor make this method less attractive due to its complexity.

6.4.7 Activity-Monitoring Completion Detection (AMCD)

Similar to sense-inverter based CSCD, this method monitors the switching activity within a combinational logic circuit to detect completion. In this case, a circuit known as an *activity monitor* (AM) (Figure 6.8), detects transitions on nodes within the logic. In the case of [118], the nodes chosen were again carry signals from selected full adders within a multiplier.

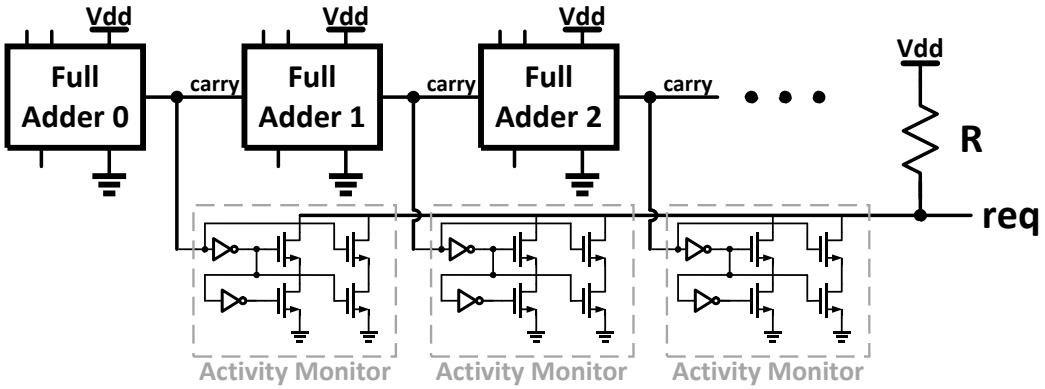


Figure 6.8: Basic configuration of *activity monitoring completion detection* within an adder structure.

Once a transition is detected (either rising or falling) the AM pulls a common *req* node low. A correctly sized pull-up resistor then pulls the signal high once the switching has finished. Depending on the expected frequency of switching activity, a capacitor may need to be added to the *req* node to ensure a false positive is avoided.

AMCD's largest advantage is that it removes a voltage droop from the logic path. This method is more challenging to implement as it uses both custom digital cells as well as analog circuits. It may also be difficult to size the pull-up resistor

correctly for each logic circuit with different switching activity factors.

6.5 Proposed Completion Detection Methods

All of the completion detection methods described above have their advantages and disadvantages. Most methods balance the trade-offs of: complexity, area/power, speed, design time, and robustness. The following proposed completion detection methods are designed to explore these trade-offs in a quantitative manner.

6.5.1 Proposed Transition Aware Completion Detection

The primary goal of this method is to provide a fine-grained completion detector that can be easily incorporated into synchronous circuit synthesis at low cost. The completion detection method, known as *Transition Aware Completion Detection* (TACD), is similar in concept to AMCD because it monitors switching activity to determine when a computation has completed. Instead of connecting activity monitors to precise internal nodes of a circuit, only the outputs to be latched need to be monitored. The system-level diagram of this method is shown in Figure 6.9(a).

TACD is comprised of a variable-delay inverter chain with a single XNOR gate per output wire (termed Transition Detectors (TDs)), as illustrated in Figure 6.9(b). The output of each XNOR is NANDed globally to produce an error/done signal for each pipeline stage. As switching activity is present for each output, the XNORs will transition low for the duration of the inverters' delay, indicating

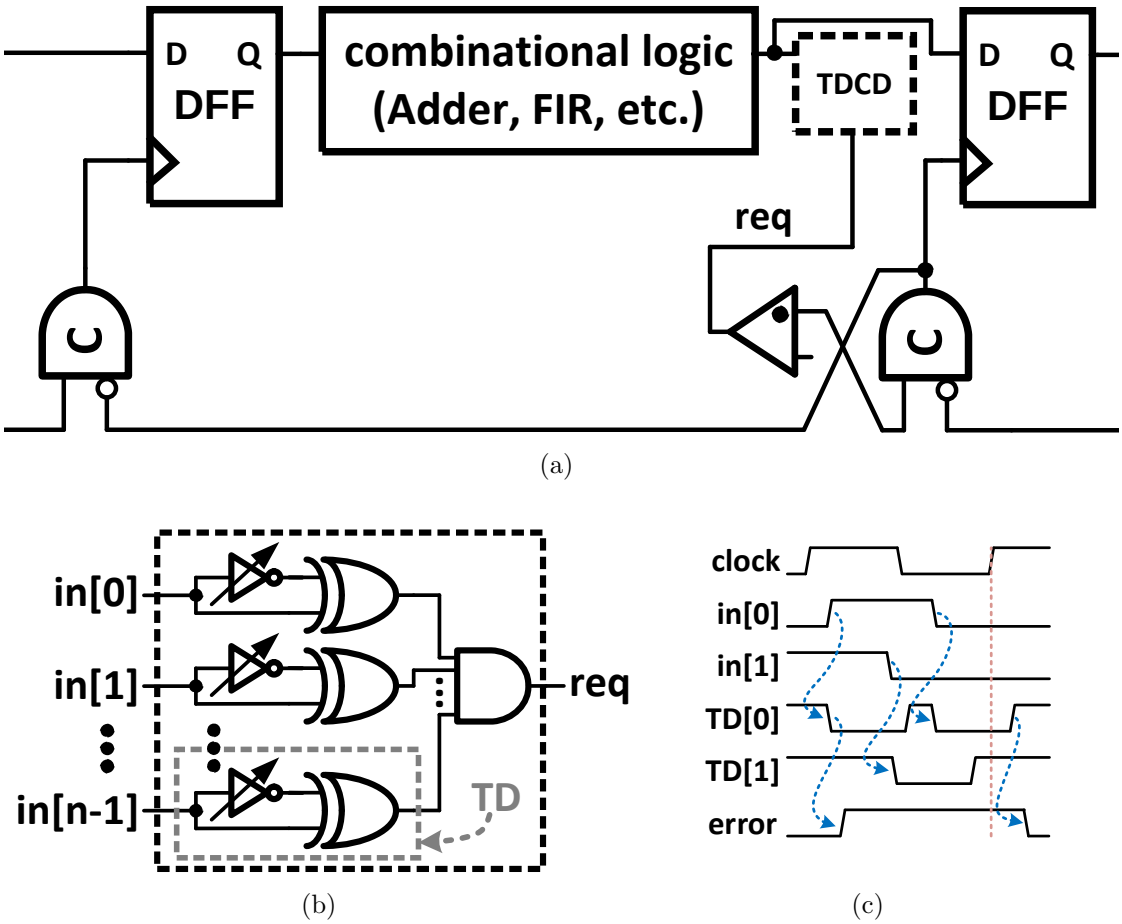


Figure 6.9: (a) TACD within a μ-pipe stage, (b) Synthesize-able TACD schematic, (c) Timing diagram of TACD.

that operation has not completed. The timing diagram for this operation can be seen in Figure 6.9(c). Note that the inverter delays need to be calibrated, depending on the amount of switching activity – not for a single output but for all the outputs combined (i.e. just long enough until another output toggles). The total inverter delay is summarized in Equation 6.1:

$$d_{inv} = d_{NAND\ tree} + \Delta_{toggle\ max} + d_{margin} \tag{6.1}$$

where $d_{NAND\ tree}$ is the worst-case delay of the NAND tree, $\Delta_{toggle\ max}$ is the worst-case delay time between any two output transitions, and d_{margin} is an added margin to account for dynamic variations.

The synthesizer can be used to determine the optimal value for d_{inv}. In the case of this work, since the adder implemented was a simple carry-lookahead architecture, it is easily determined that $\Delta_{toggle\ max}$ results in the longest carry propagation path. In the presence of a synthesized circuit that generates glitches, the equation becomes slightly more complicated in that any given Δ_{toggle} must be less than the time between either another glitch or a legitimate output toggle. Glitches can be removed in some cases by adding a prime implicant. However, this requirement may be hard to ensure with more complicated logic.

In order to improve the performance of TACD, TDs can be added at strategic points within the logic. In the case of a CLA adder, a TD can be connected to the carry-out of each lookahead unit. This allows each inverter delay to be smaller, requiring less area for each TD. Even though there are more TDs, the total area increase is minimal for relatively complex digital logic blocks.

6.5.2 Proposed Pseudo-Asynchronous CSCD Method

In order to draw comparisons with other completion detection methods, a synchronous CSCD scheme has been modified for asynchronous operation. This

method consists of a clocked analog comparator measuring the voltage droop across a large PMOS header transistor. It has been modified for asynchronous operation via a feedback of two NAND gates. The schematic of this is shown in Figure 6.10. Once a conversion finishes, the comparator effectively generates its own clock and continues to clock itself until the current consumption goes below its calibrated offset. The comparator is restarted with the μ-pipe stage's *req* signal controlling the reset input. This feature allows the sensor to save energy while not switching when no comparison needs to be performed.

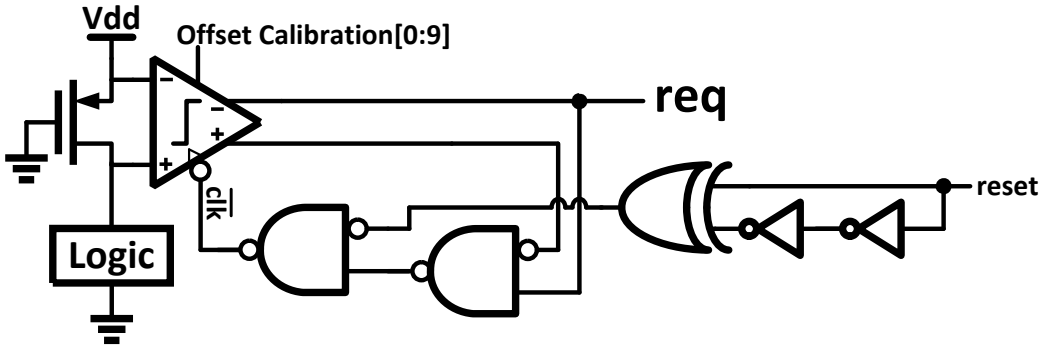


Figure 6.10: Synchronous CSCD method modified for asynchronous operation.

6.6 Compared Results

The two completion detection methods above, along with a simple matched delay line (MDL) are designed, simulated, and compared in an IBM 90nm CMOS process. Each method was used to detect the completion of a synthesized 16-bit carry-save multiplier and asynchronous control signals were emulated to ensure the methods would generate correct μ-pipe communication signals. Table 6.1 shows

Table 6.1: Comparison of different completion detection methods.

	Worst	**Average**	**Best**
TACD Delay	*1.6ns*	*1.1ns*	*0.36ns*
TACD Energy (TDs)	$1.074pJ$	$0.762pJ$	$0.012pJ$
TACD Energy (AND)	$0.118pJ$	$0.074pJ$	$0.013pJ$
TACD Multiplier Energy	$1.812pJ$	$2.124pJ$	$0.018pJ$
TACD Energy Overhead	39%	28%	57%
TACD Total EDP	**4.8**zJs	**3.2**zJs	**0.015**zJs
MDL Delay	*1.4ns*	*1.4ns*	*1.4ns*
MDL Energy	$0.119pJ$	$0.119pJ$	$0.119pJ$
MDL Multiplier Energy	$1.74pJ$	$2.11pJ$	$0.023pJ$
MDL Energy Overhead	6%	5%	83%
MDL Total EDP	**2.6**zJs	**3.1**zJs	**0.2**zJs
CSCD Delay	*1.8ns*	*1.84ns*	*0.18ns*
CSCD Energy	$0.117pJ$	$0.138pJ$	$0.009pJ$
CSCD Multiplier Energy	$1.15pJ$	$1.41pJ$	$0.0095pJ$
CSCD Energy Overhead	9.2%	8.9%	1.5%
CSCD Total EDP	**2.3**zJs	**2.8**zJs	**0.003**zJs
TACD Area Overhead	$853\mu m^2$ (16%)		
MDL Area Overhead	$191\mu m^2$ (4.1%)		
CSCD Area Overhead	$816\mu m^2$ (15.5%)		

the simulated delay, energy and area of each method. Three cases were tested: (1) *Worst-case* multiply vectors from the synthesizer, (2) *Average-case* multiply vectors generated randomly, (3) *Best-case* multiply vectors in the form of 0-times-0 to 1-times-1.

TACD performs well when tuned correctly. The effective delays of this method are lower than the other methods tested on average. However, because the inverter delays have to be tuned so large to unsure no hazards there is a delay in the final *req* output (Figure 6.11).

The MDL performs as expected. When fixing the delay just above the worst-case at 1.4ns the MDL clearly has downsides when the delay is fast.

Most of the CSCD's area is due to the large offset-calibration circuitry required to ensure correct operation. One down side of this method is the requirement of the droop on the virtual supply. In the case of a 16-bit multiplier, the droop has to be quite large in order to ensure that all droops are properly detected, both small and large. This large droop has a negative affect on delay as shown in the bottom half of Figure 6.11. It also results in the deceptively low energy of this implementation. Furthermore, it can be seen that the sensing voltage takes several

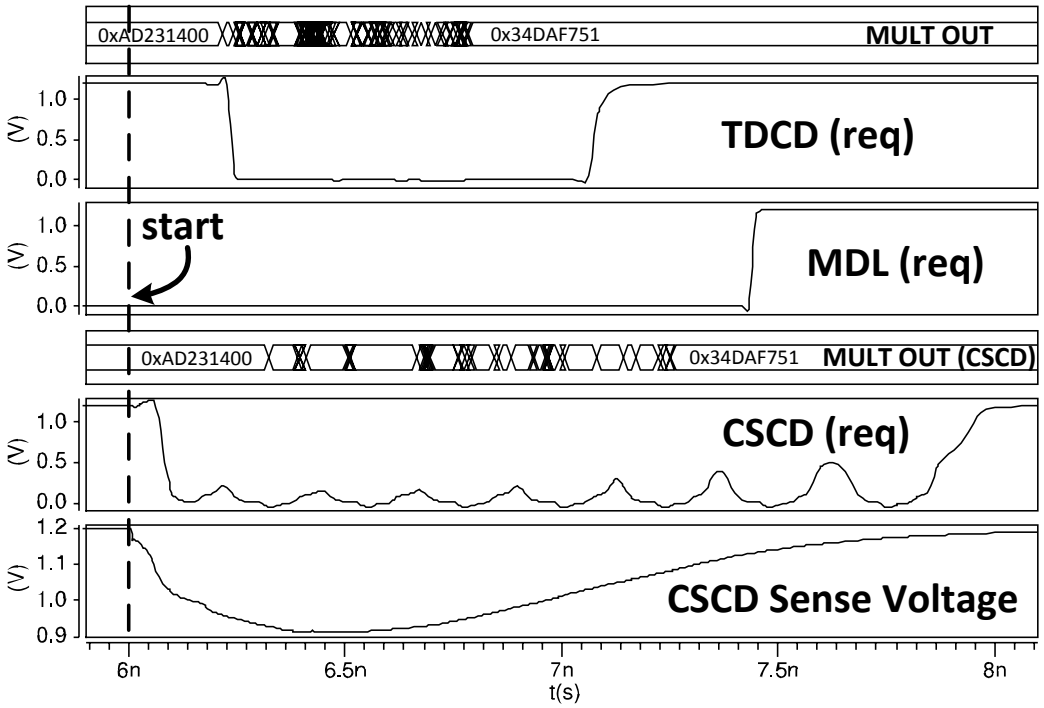


Figure 6.11: Simulation of TACD, MDL, and CSCD with average-case operands into a 16-bit carry-save multiplier.

hundred picoseconds to settle even after the output finishes toggling, making the *req* signal slower than the worst-case delay. Pairing CSCD and MDL could solve this problem. Future work needs to be done to determine the sensor granularity and how many sensors need to be used.

In the last row of each section of Table 6.1 the Energy Delay Product (EDP) calculations show that as the delay of the multiplier approaches its average case, CSCD outperforms both TACD and MDL. However, in the configuration presented, CSCD provides almost no speedup opportunities which make TACD a better candidate when throughput is important. Furthermore, it should be noted that the synthesize-ability and ease of integration of TACD make it a much more promising candidate for future asynchronous systems.

6.7 Conclusions

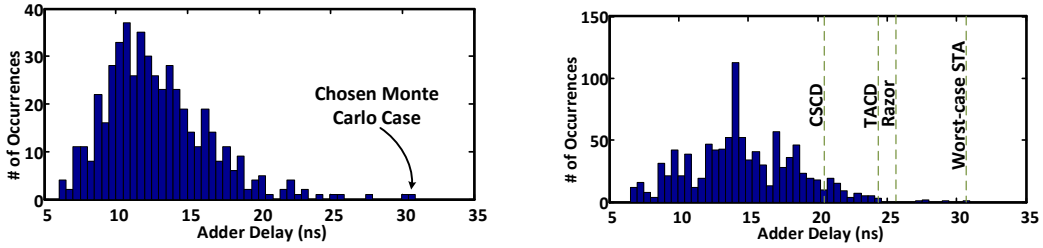
This chapter presented comparisons of all the relevant asynchronous completion detection methods to date. Along with these, two new completion detection methods have been compared: TACD, and a pseudo-asynchronous version of CSCD. Simulations have shown that with the addition of generic completion detection circuits to an asynchronous micropipeline, it is very difficult to get an energy reduction compared to traditional synchronous circuits aside from the removal of the global clock network. In that light, for asynchronous circuits to be successful, designers must be willing to design completion detection schemes that are architecture aware in order to make the most of asynchronous operation.

Chapter 7: Synchronous Completion Detection

7.1 Introduction

Asynchronous circuits clearly have a large value in systems where delays are unpredictable. However, their design complexity still makes them unattractive to most of the design community. In order to make synchronous circuits just as reliable as asynchronous circuits it could be possible to combine the best of both worlds.

In this chapter, two ideas for ensuring timing-resilient synchronous circuit operation at a lower supply voltage are proposed based on the methods proposed in Chapter 6. First, a study on the characteristics of timing variations in near-threshold are presented. Next, a discussion of conventional methods for circuit-level timing error detection, and their limitations in the near/sub-threshold domain will be discussed. Then, two asynchronous methods adapted to synchronous pipelines will be presented, illustrating their effectiveness in the near-threshold voltage regime. Finally, an experimental simulation setup will be presented and discussed, comparing the results with previous works and illustrating the potential throughput advantages of the proposed designs.



(a) (b)

Figure 7.1: Histograms of (a) Monte Carlo chip-to-chip delay of the STA and (b) delay of changing FIR filter data on a 16-bit adder with error-detection speeds marked.

7.2 Variation Study

In order to explore the effects of process variations in the NTV regime, a 16-bit Carry-Lookahead (CLA) adder was synthesized in a 90nm CMOS process. Figure 7.1(a) shows the histogram of a 500-point Monte-Carlo simulation of the CLA adder, where the inputs are the worst-case static timing analysis (STA) vectors determined by the synthesizer. The figure shows a large standard deviation in delay while operating at a near-threshold voltage of 0.5V. On top of process variation-induced timing uncertainty, is input vector variation. Figure 7.1(b) shows the simulated delays of the same adder at one particular Monte-Carlo case, where the input vectors are supplied from the outputs of a FIR filter. Further, these vector-to-vector timing variations worsen as the circuits operate deeper in the near-threshold regime.

Figure 7.2 shows the potential speedup that can be ideally achieved with the ability to detect all timing errors. For this simulation, the worst-performing Monte

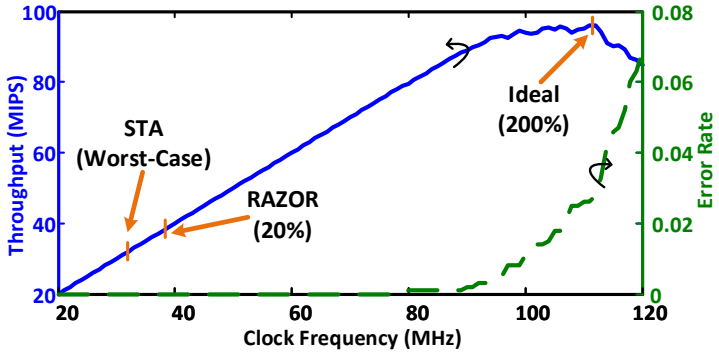


Figure 7.2: Potential throughput improvement with ideal error detection

Carlo case from the 500 simulated was chosen, resulting in a worst-case clock speed of $32MHz$ (as opposed to a best-case speed of $166MHz$). Next, 1000 add-vectors from a low-pass FIR filter using electroencephalography (EEG) data were extracted from a Matlab simulation, and then simulated with a 16-bit carry-lookahead adder operating in near-threshold. Using either the micro-rollback or counterflow pipeline error-recovery methods [119], assuming that an ideal error detection method exists (one that can perfectly detect all errors at any clock speed), the potential speedup can be as much as 200%.

These HSPICE simulation results suggest that circuits operating in the near-threshold regime cannot afford to be margined for the worst-case while still ensuring error-free operation with predictable yield, in regards to both throughput and energy-efficiency.

7.3 Proposed Error Detection Methods

The following section introduces two completion-based error detection techniques that can improve throughput beyond the limitations of conventional Razor circuits. The first, Transition Aware Completion Detection (TACD), is a fully synthesizable method similar to Razor circuits. The second, known as Current Sensing Completion Detection (CSCD), is an analog approach that uses a current sensor to monitor the supply droop to detect errors. Both methods were introduced for asynchronous operation in Chapter 6.

7.3.1 Transition Detecting

Conventional Razor circuits detect errors due to output changes after the clock edge. The proposed TACD method detects errors based not on output value correctness but on output value transitions, and not after computation completion but during the computation. A detailed description of TACD is presented in Section 6.5.1.

7.3.1.1 Detector Resiliency

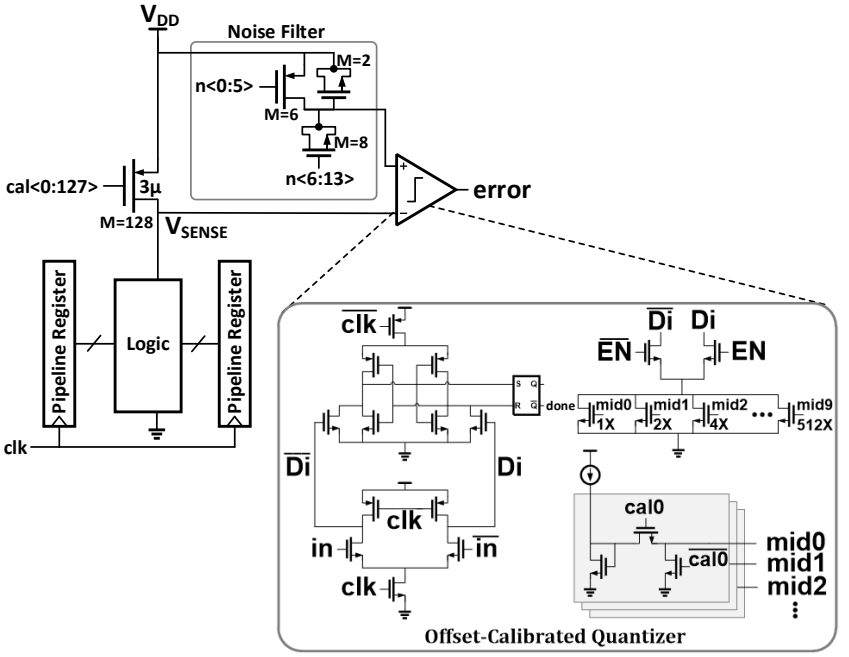
Because the TACD error-detector will be used in the same highly-variable near-threshold environment as the combinational logic, it must be designed for error-free operation. Because the detector uses only simple logic parts, it can be easily tuned for near-threshold operation, and the detector as a whole can continue to

operate in the presence of variations. d_{inv} can be determined using simple delay tests, and can therefore be easily tuned using off-line calibration after post-silicon fabrication. For example, off-line delay-path tuning was previously proposed with tunable replica circuits (TRCs) [73].

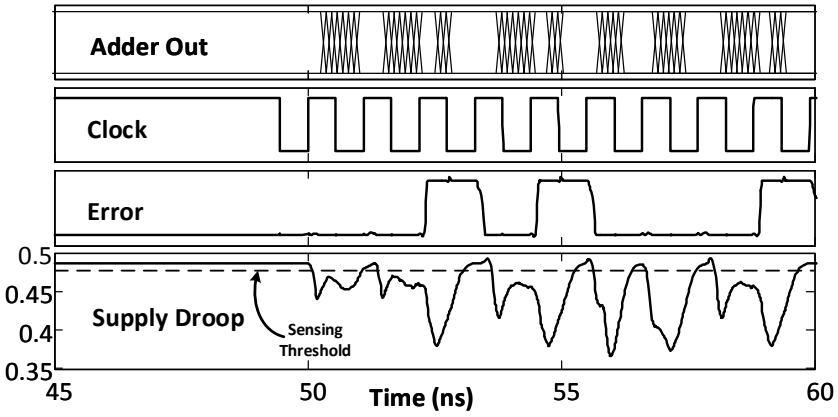
7.3.2 Current Sensing Completion Detection

CSCD [116] consists of an analog sensor that senses the current through a group of combinational logic via a resistor or power-gate transistor. When a system begins to compute on new input vectors, the logic's current consumption increases. When this current consumption abates to a steady state, the computation has completed. While these types of circuits have been evaluated for asynchronous operation (such as the design presented in Section 6.5.2), they have not been applied specifically to synchronous systems operating in near-threshold where timing variations are a critical concern.

The proposed CSCD method consists of a clocked, offset-programmable, dynamic sense-amplifier that measures the voltage droop across a large PMOS power-gate transistor (Figure 7.3(a)). As power gates are becoming ubiquitous in modern digital designs [120], they do not contribute to the area overhead for this work. Because the current consumption only needs to be measured at the clock edge, there is no need to use a continuously-monitoring sensor, like that in asynchronous versions of CSCD [116]. Figure 7.3(b) shows a sample SPICE waveform of the virtual supply droop and resulting error detection at the clock edge.



(a)



(b)

Figure 7.3: (a) CSCD schematic, (b) CSCD timing diagram.

7.3.2.1 Sensor Resiliency

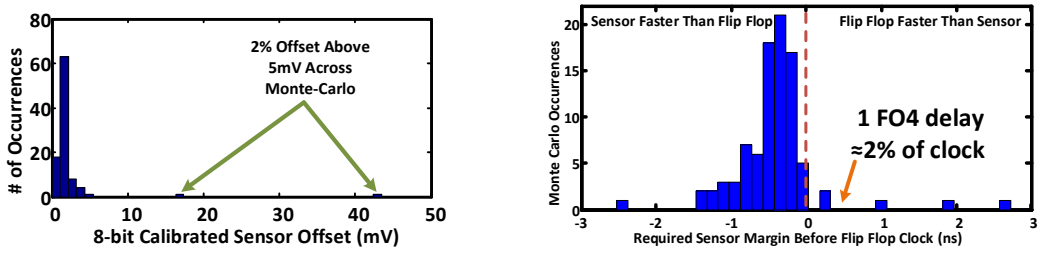
The CSCD sensor will operate in conditions that are more harsh than typical super-threshold operation. These conditions include process variations, slow NTV operation, temperature variations, small virtual supply droops (affecting minimum input sensitivity), and supply noise.

In order to combat the exacerbated process variations that occur in the NTV regime, a well-known offset calibration scheme in the form of current steering is chosen [121], as shown in Figure 7.3(a). A current-steering DAC along with a simple one-time calibration procedure is used to set the residual offset below 5mV under most extreme variations, including near-threshold operation. To perform the calibration, one tail of the sense-amplifier is chosen and the calibration bits of the current DAC are incremented once for each calibration cycle. Once the sensor reports the error signal the calibration is subtracted to set the sensor threshold just below the settling voltage of the supply. This calibration scheme can be extended to combat slow-changing variations like temperature by performing live in-situ calibrations periodically.

Figure 7.4(a) shows a histogram of the calibrated offset using 8-bits of calibration of the CSCD sensor. Only 2 of the 100 cases have large offsets above 5mV. Larger offsets can be compensated for by increasing the dynamic range of the reference currents into the quantizer.

The CSCD sensor must make a quantization before a new set of data is clocked into the pipeline stage. This is analogous to the min-path race condition problem

that exists for Razor-based systems. Figure 7.4(b) shows a plot of the sensor conversion speed relative to flip-flop D-Q delay across 100 Monte Carlo points. The majority of cases result in faster conversion speed than the D-Q delay. If timing is a concern, small delay buffers can be added between the sensor and the flip-flop clocks.



(a) (b)

Figure 7.4: (a) Offset calibration of CSCD across 100 Monte Carlo simulations, (b) Speed of sensor stays relatively fast in NTV regime.

One important concern with all CSCD methods is sensing margin. The voltage droop on the virtual supply needs to be large enough to allow the detection of computational errors, but small enough to mitigate a negative impact on performance, due to the large voltage drop on the virtual supply. Figure 7.5 plots both logic speed and sensing margin versus power gate size. It can be seen that this voltage drop can be quite large (˜40mV) without negatively impacting speed. For this work, a 100μm power gate was chosen not only for its droop and speed characteristics, but also due to its smaller impact on area, compared with a larger power gate that provides a minimal speed improvement. The power-gating transistor is parallelized and digitally controlled (30 parallel header-PMOS transistors), thus

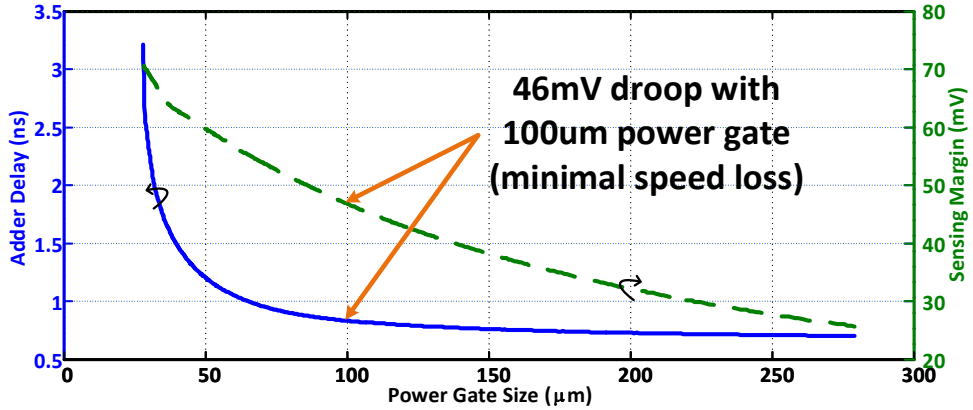


Figure 7.5: Power gate sizing has very minimal affect on speed while maintaining a reasonable sensing margin.

allowing for programmable amounts of voltage droop across process skews. It is also important to note that this method is strictly limited to supply voltages above or near the near-threshold operating point, as it will only work when the switching current is discernible from the leakage current. This limits this sensor to process nodes with lower leakage and possibly higher operating voltages than the optimal near-threshold voltage.

Another major concern for this type of circuit is noise. Measuring a small voltage drop across a header transistor can be extremely difficult, especially with supply noise and other sources of noise. Hence, a proposed differential configuration of the sensor can cancel common-mode noise at the inputs, assuming both inputs experience the same noise filtering. In order to make sure the two differential inputs are correctly correlated with any power-supply noise, a replica RC-matching circuit was designed. Shown in Figure 7.3(a), the circuit consists

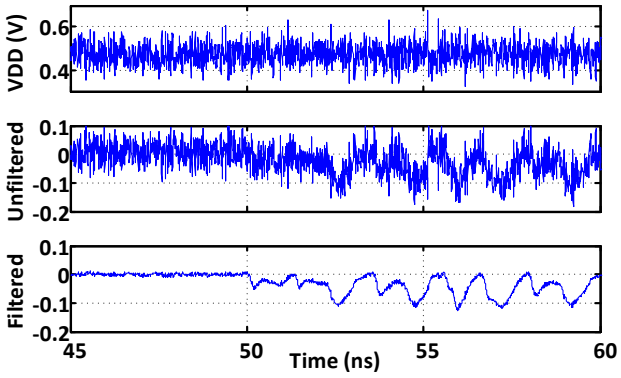


Figure 7.6: Input referred supply noise before and after calibrating CSCD sensor's RC noise filter.

of digitally-controlled resistors and capacitors that can be tuned post fabrication to match the RC characteristics of the power gate and logic. Simulations show a successful power-supply noise reduction of 20x (from 200mVpp to under 5mVpp) at the differential inputs after proper digital calibration (Figure 7.6).

7.4 Results

In order to quantify the robustness of the two near-threshold error detection methods, simulations were performed at a near-threshold voltage of 500mV, comparing Razor, TACD, and CSCD. Using the worst-case static variations utilized in section 7.2, HSPICE simulations were carried out on all three error-detection methods.

The simulations were designed to find the fastest clock speed at which an error could be detected. Hence, the 16-bit CLA adder was simulated across Monte Carlo process variations, on 1000 input vectors extracted from the EEG FIR filter. To

ensure simulation coherency between the three different error-detection methods, the outputs and current consumption of the adder were first extracted and then used as input stimuli for separate simulations of each error detector. Each error-detection method was simulated to find its optimal operating speed, given the simulated delays of the 1000 vectors.

7.4.1 Razor Results

In the case of Razor, the fastest clock speed can only increase 20% faster than the STA, whereas TACD and CSCD can be clocked much faster. For Razor, given the limited input data simulated, no errors were generated because all delays were 20% faster than the worst case. This implies that Razor does yield a throughput of 20%, but it is clear that it could benefit from an even faster clock speed. Hence, the choice of these 1000 input vectors may not have stressed the worst-case logic delays, which sets the delay of the Razor clock, and therefore the best possible clock speedup.

7.4.2 TACD Results

Because TACD essentially lengthens the datapath by adding inverter delays before the error signal, its throughput exceeds Razor only marginally. Figure 7.7 shows the simulated throughput using TACD. Because of the finite delay of the TDs, many residual error signals are flagged, resulting in a 29% improvement in

throughput with TACD, after considering the error rollback delay overhead. Note that TACD does not require a min/max logic delay guarantee within 20%, as required with Razor, but does require an initial off-line calibration procedure for calibrating the inverter buffer delays.

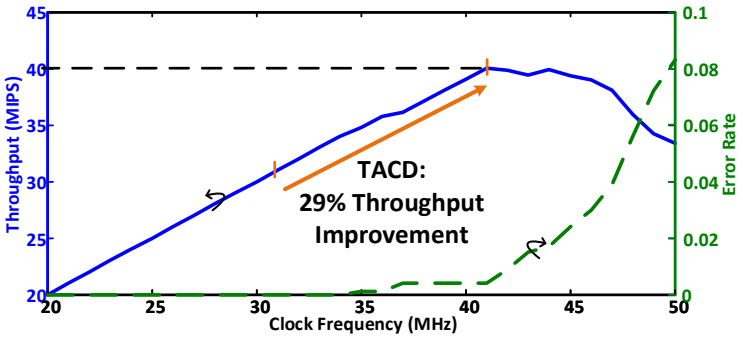


Figure 7.7: Simulated throughput of TACD.

7.4.3 CSCD Results

Simulated throughput for CSCD in the near-threshold regime shows significant improvements over both Razor and TACD. Since CSCD does not add any delays to the datapath, its throughput nearly triples Razor's average performance improvement (56%). As shown in Figure 7.8, the throughput saturates due to errors generated by the finite settling time of the virtual supply droop and small delay increase associated with the droop, as seen in Figure 7.5.

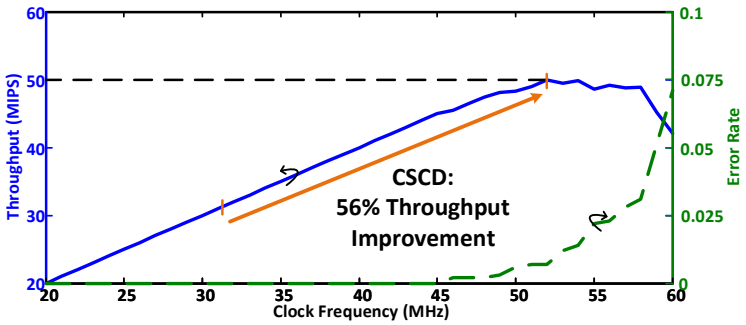


Figure 7.8: Simulated throughput of CSCD

7.4.4 Energy, Area and Complexity

Along with the throughput improvements, Table 6.1 compares the energy and area overheads of the error-detection methods. Error recovery methods [119] are neglected because their overhead is independent of the detection method used. At $976.5fJ$, Razor's energy consumption is dominated by the energy of the shadow latch. The minimum-delay buffer insertion contributes 21% to the overall energy increase.

Exhibiting a significantly less area footprint to Razor (mainly due to the large overhead of Razor's inserted min-delay buffers), TACD consumes energy similar to Razor.

CSCD, with a capacitance equivalent to one large logic gate, consumes the least amount of energy. Furthermore, because it is clocked only once per cycle, CSCD's dynamic contribution to the energy is much smaller than the other two methods. However, it does require more area than TACD, due to the large offset and calibration process required to ensure correct near-threshold operation.

Table 7.1: Comparison of error-detection methods.

	No Detection	**Razor**	**TACD**	**CSCD**
Average Energy	192.5fJ/comp	976.5fJ/comp	910.3fJ/comp	195.9fJ/comp
Throughput (% Increase)	32MIPS (0%)	38.4MIPS (20%)	41.2MIPS (29%)	49.9MIPS (56%)
Area	15495μm^2	24080μm^2	15796μm^2	16005μm^2
Complexity	NA	Medium	Medium	High
Near-Threshold Variation Adaptability	None	None	Tunable TDs	Robust Calibration

One other key factor contrasting all three of these designs is complexity. First, it has not been proven that the minimum-delay buffer insertion required for Razor's operation will scale correctly for near-threshold operation. Therefore, although any synthesizable form of Razor may be relatively simple to implement, Razor circuits are difficult to guarantee error-free operation across instances of extreme variations. TACD, being fully synthesizable, is easy to implement with logic. However, improving its performance using architecture-dependent techniques and post-fabrication tuning of the transition detectors increases TACD's implementation complexity. This is especially true for ensuring glitch-free operation of the error signal across process corners. CSCD, exhibiting the best throughput improvements, is also the most complex to implement. Designing and adding the analog sensor to a digital circuit will be challenging, such as the post-fabrication calibration required for proper operation across process and supply voltage variations.

7.5 Conclusion

This chapter introduces two new approaches to circuit-level timing error detection. Transition-Aware Completion Detection (TACD) observes the activity of the outputs of the combinational logic using an XNOR gate and a variable-delay inverter chain, which is calibrated based upon the amount of switching activity that exists in the logic. The technique of Current-Sensing Completion Detection (CSCD) to the near-threshold domain was also introduced. CSCD consists of a current sensor that bases its completion/error signal on the current consumption profile of combinational logic across a power gate. These methods were compared to the well known Razor error-detection technique operating in near-threshold. Comprehensive HSPICE simulations show that both TACD and CSCD outperform Razor in throughput, area, and energy and they also provide a good basis for future work in near-threshold error detection.

Chapter 8: Current Sensing Completion Detection Test Chip

8.1 Introduction

The CSCD design described in Chapter 7 is a very promising solution for robust and reliable near-threshold error detection. In order to prove it's effectiveness a test microchip was designed in a commodity $65nm$ CMOS process. This chapter describes the chip design and measurement results as supply voltage is scaled. After a brief discussion of previous work, the rest of this chapter goes on to describe the test chip in detail. The design is thoroughly analyzed, measurement results are shown, proving the effectiveness of a real implementation of CSCD, and conclusions are drawn.

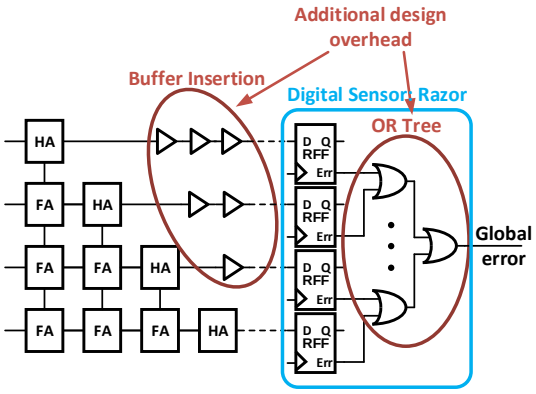
8.2 Previous Work Comparison

In order to better illustrate the potential benefits of CSCD, Fig. 8.1 shows the overheads associated with Razor, TRC, and CSCD. Table 8.1 then draws a qualitative comparison of the three designs. The Razor technique (Fig. 8.1(a)) has three

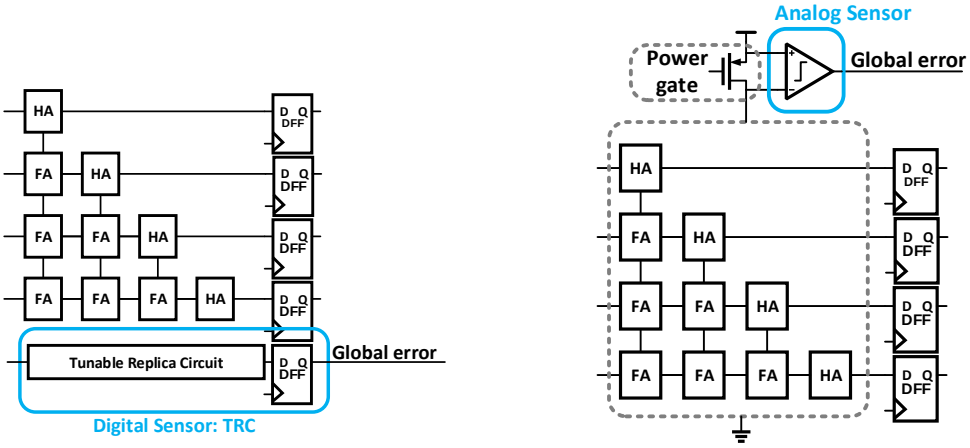
Table 8.1: Comparison of existing error-detection method with measured test chip.

	Razor [5]	**TRC [73]**	**This Work**
Detection Method	Shadow Latch	Delay Line	Current Sensing
Area Overhead	6.9% (with 17% coverage)	2.2%	1-2% / sensor
Detection Window	13-20%	100%	100%
Speedup Potential	13-20%	None	Not Limited
Main Design Challenge	Hold Time	Delay Matching	Noise, Sensing Margin
Primary Limitation	Detection Window	False Positives	False Positives
Low Voltage Operation	Large Buffer Overhead	Max Path Calibration	Droop Calibration

main components that need to be designed for it to work properly: min buffer insertion, razor flip flops, and an OR-tree. As discussed in Chapter 3, the min-buffer can be difficult to design for when attempting to make all paths delays to be as matched as possible. This is especially true when operating at low voltages. All of the overheads associated with Razor expand with data width. TRC (Fig. 8.1(b)) does have a distinct advantage over Razor in that the overhead is small but the delay line can only replicate the worst-case delay. Meaning, even if a computation finishes faster than the max path the TRC will still flag an error. CSCD's overhead complexity (Fig. 8.1(c)) is similar to TRC, yet with the detection ability of Razor. The only overhead required per group of logic is a single sensor. This completely eliminates the requirement for strict timing constraints and precision digital design (i.e. no min-path buffer insertion). It also allows for an unlimited timing speculation window since the delay of any path can be detection, not just the worst cast.



(a)



(b) (c)

Figure 8.1: Conceptual demonstration of: (a) Razor technique; (b) TRC technique; (c) Proposed current sensing completion detection technique.

8.3 CSCD Test Chip Design

The test chip was designed using a mixed-signal design flow in $65nm$ CMOS. Two SIMD cores [122] were custom designed using an all-digital design flow identical to what would be seen in industry. The cores were written in Verilog, synthesized in Design Vision, and place and route was performed in SOC Encounter. The Cores consist of a 5-stage pipeline with power gates connected between the supply rail and the virtual supply of each stage. Fig. 8.2 shows a basic block diagram of each core.

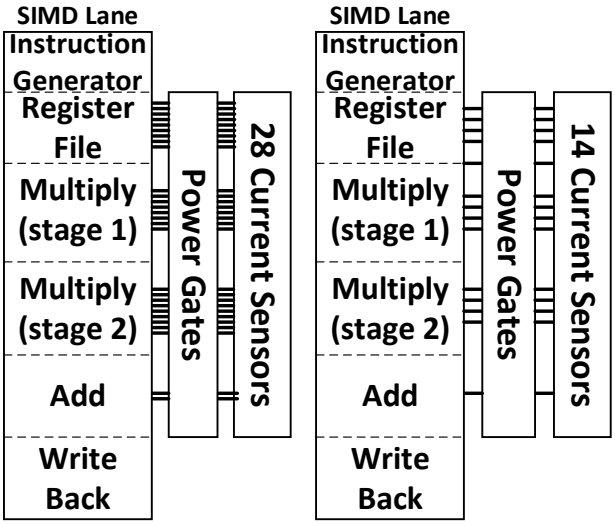


Figure 8.2: Block diagram of test chip with sensors connected to SIMD lanes.

8.3.1 SIMD Pipeline Design

8.3.1.1 Instruction Generator

The first instruction fetch stage (typical to modern pipeline design) has been replaced by an Instruction Generator (IG) stage. This stage is used to test the later stages by generating a predetermined set of instructions that exercise a worst-case use profile of the pipeline. As shown in Fig. 8.3(a) this stage consists of a linear feedback shift register (LFSR), counter, and instruction generation logic. After being reset, the LFSR starts generating 16-bit pseudo-random numbers. The instruction logic takes these numbers and generates an instruction that it knows will compute without any memory access hazards. It also ensures that all Register File (RF) words are written to by the 31st clock cycle. Once the counter reaches 32 the instruction logic starts to generate instructions that incorporate hazard-free memory read operations.

8.3.1.2 Register File

The RF stage is a flip-flop based synthesized 32-byte register file that is capable of single-port read and single-port write simultaneously (Fig. 8.3(b)). Each word is 32 bits long with 8 total words. One address decoder is used for reading, eliminating the need for clocking during a read operation. A write address decoder simply controls a clock gate for each word. The current sensors for the 28-sensor version of the pipeline have one sensor per logic word. The 14-sensor version has one

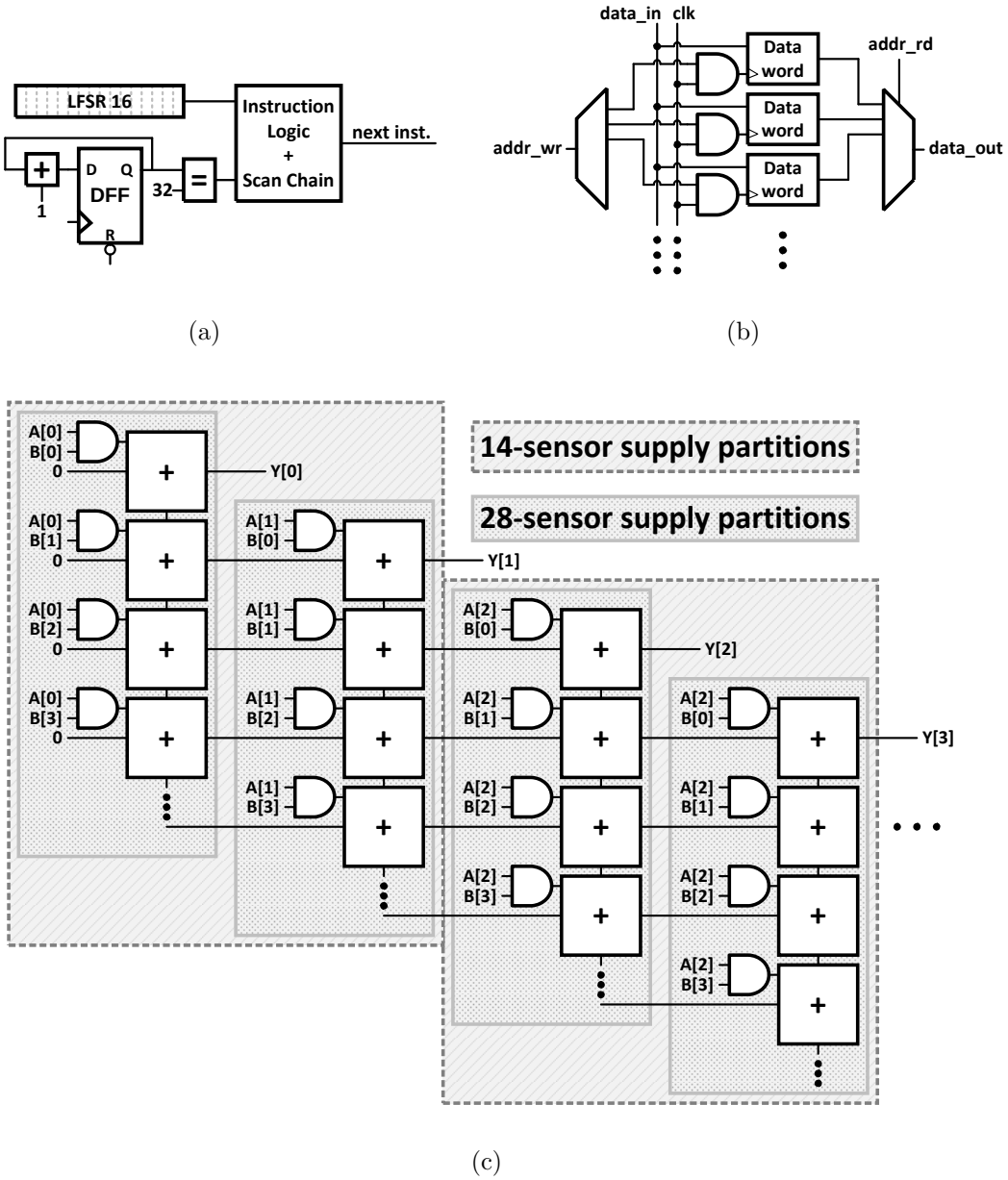
sensor per two logic words. Multiplexer and decoder logic is divided evenly among the sensors for each version.

8.3.1.3 Multiplier

The 16-bit multiplier in this design is a simple array multiplier with 16 stages. The pipeline is split up into two stages with 8 array-stages in each. Fig. 8.3(c) illustrates the current sensor breakdown for the multiplier. As can be seen, for 28 sensors each 16-bit slice has its own sensor. For 14 sensors, every two slices have their own sensor. The multiplier is partitioned such that an equal worst-case delay for each stage is maintained.

8.3.1.4 Adder

The next stage of the pipeline is a 32-bit adder that can either add the result of the multiplication with a constant or a word from memory. The adder consists of a simple ripple-carry topology. For the 28-sensor pipeline two sensors are used on the adder, evenly divided in half through the critical path. For the 14-sensor version only one sensor is used for the whole adder.



(a) (b)

(c)

Figure 8.3: Detailed architecture of pipeline stages: (a) Instruction Generator; (b) Register File; (c) Pipelined Array Multiplier (stage 1).

8.3.1.5 Write Back

The final stage of the pipeline contains no logic and therefor has no sensors. The write back (WB) stage passes the result of the adder either back through the adder, multiplier, or RF to be written. The logic associated with each of these operations is contained in its respective stage.

8.3.2 Current Sensor Design

The current sensor in this chip is designed similar to the sensor described in Section 7.3.2. Some modifications have been made to improve operation in the $65nm$ process it was designed in, as well as the addition of testing features. Namely: a test input mux, noise replica filter modifications, power gate sizing changes, and the addition of a 'mincal' transistor.

The offset-calibrated quantizer shown in Fig. 7.3(a) is the same as what is used in this chip except there are 8 offset calibration bits instead of 9. It was found through simulation that 8 bits were more than what was needed to ensure that the offset could be calibrated passed $0mV$. As can be seen in Fig. 8.4 a test mux was added to enable analog test voltages to be used to characterize and calibrate the sensors off-line. These muxes can also take the analog V_{SENSE} voltages from a single sensor to off chip, in order to verify the effects of power gate sizing on the virtual supply.

It was found through simulation that the component of the noise replica filter described in Section 7.3.2 that had the most impact on noise rejection was bits

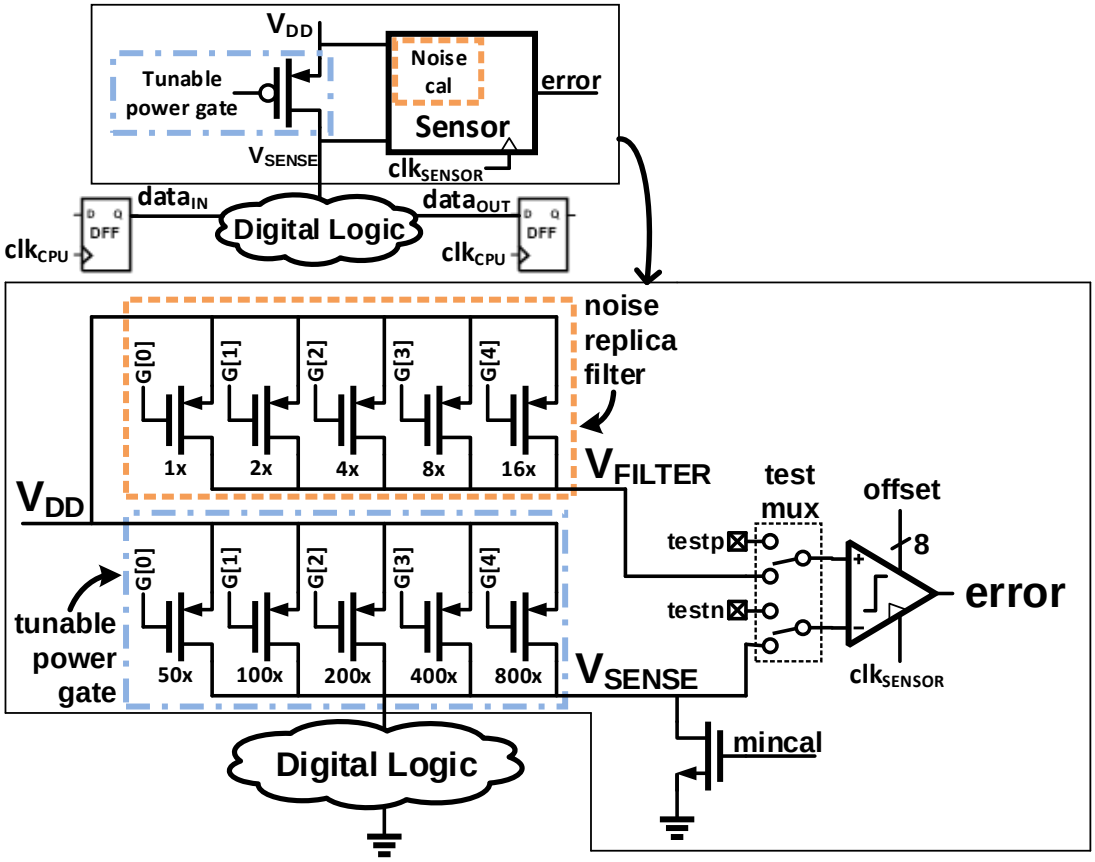


Figure 8.4: Schematic of current sensor used in test chip.

$n < 0 : 5 >$. The capacitance's contribution to common mode noise rejection was minimal. Removing it from the design had almost no negative impact and made tuning much easier. As can be seen in Fig. 8.4, a 5-bit noise replica circuit was chosen for two reasons. First, reducing the number of bits from 14 to 5, reduces the requirements on the calibration circuitry complexity. Secondly, 5 bits is the same amount of tuning range as the power gate, potentially allowing for the replica filter and power gate to share the same calibration bits/circuitry to further reduce design complexity.

Based on simulations in $65nm$ with the SIMD design that was to be placed on the chip, the power gate sizing requirements were refined. A 5-bit binary weighted power gate was determined to have more than enough tuning range to allow for a suitable sensing margin across all process corners. The same power gate was used for each sensor. This set the minimum and maximum requirements for the amount of logic each sensor could protect which is part of the reason overheads of 14- and 28-sensor configurations were chosen. If a custom power gate was designed for each pipeline stage, instead of for each sensor the design could be optimized further. The primary limitation of the current power gate configuration is the LSB, or minimum size which, as will be described in Section 8.4, limits the minimum supply voltage the droop can still be detected with.

Another addition to this sensor design is the 'mincal' transistor. This transistor is designed to have very low leakage and little response to process variations (via long channel length and up-sizing). When activated, the transistor pulls the worst-case (smallest) amount of current that a single logic gate would pull when

switching. This allows for the sensor's threshold to be calibrated to the virtual supply voltage at that state. This dramatically simplifies calibration requirements for each sensor and reduces overall calibration time.

8.3.2.1 Sensor Calibration Procedure

The current sensor and test chip were designed with sensor calibration in mind. Great effort was put into ensuring that calibration could be carried out as efficiently as possible with simple and limited hardware overhead. Fig. 8.5 shows a flow chart of the proposed calibration procedure for this sensor design. The procedure is generally divided into two steps: offset cancellation and power gate sizing.

The first step is to maximize the power gate size to ensure the inputs to the sensor are as close to each other as possible (minimizing any leakage-induced droop in the V_{SENSE} node). The sensor is then clocked to determine its initial offset without any current steering enabled. Depending on the sensor result, current steering will be enabled on either the non-inverting side (Cal) or inverting side (Cal_b). Once this is done, the sensor will be clocked a maximum of 8 more times, one for each binary bit of the DAC. Because the DAC is binary weighted and not thermometer weighted the calibration time is dramatically cut down to $\mathrm{O}\big(log(n)\big)$ instead of $\mathrm{O}\big(n\big)$ complexity. Once the sensor has enough offset calibration to switch it's value the offset is subtracted to make sure the sensor's steady state is such that it is reporting no error.

Next, the power gate must be sized properly to detect the current of a single

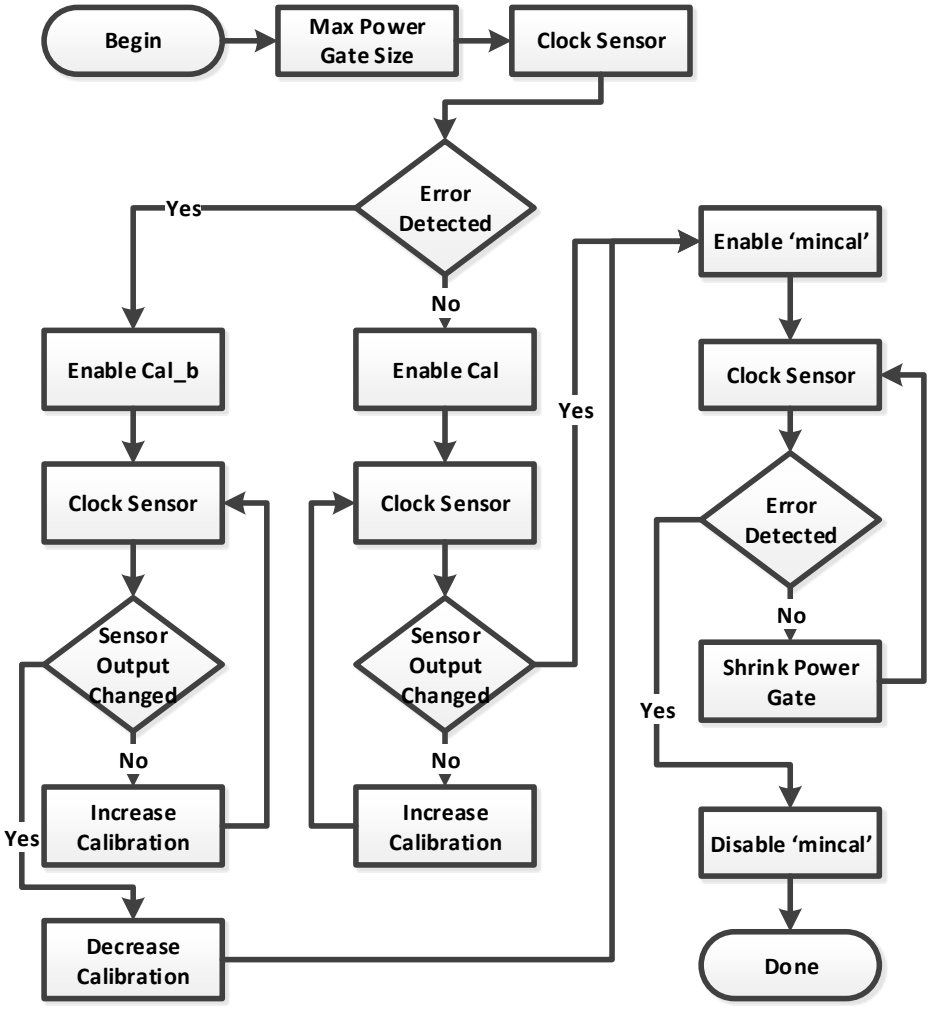


Figure 8.5: Flow chart of calibration procedure.

logic gate switching. Just as the offset was tuned with the DAC, the power gate too only requires O $log(n)$ complexity as it is binary weighted. The 'mincal' transistor is enabled, the sensor is clocked, and the power gate is tuned smaller. Once the power gate gets small enough to trip the sensor with the current draw of the 'mincal' transistor, calibration is done and 'mincal' is disabled.

8.3.3 Design for Test Features

The chip (Fig. 8.6) has many features allowing for easy testing of the pipelines and current sensors. These features were specifically designed to allow for easy testing and debugging while not impeding the operation of any of the critical circuits.

A current bias generation circuit was designed to allow a programmable current to be sent to each individual sensor's current-steering DAC. Two analog input pads can be used to put any arbitrary analog voltage into the test inputs of each sensor. All sensors share the same two reference voltages. This can also be used to pass an analog voltage from one of the sensor's inputs off chip.

A dual-clock scan chain is used to take digital data in and out of the chip. This can control every calibration bit of every sensor on the chip. It can also control clock gates that enable/disable any single sensor. Output data from the sensors can be taken out of the chip as well. Every pipeline register in the SIMD cores can be loaded and/or scanned out independent of the sensors.

Finally, a triple modular redundant (TMR) error counter and locator unit with a high-speed replica pipeline can be used for more advanced testing. The

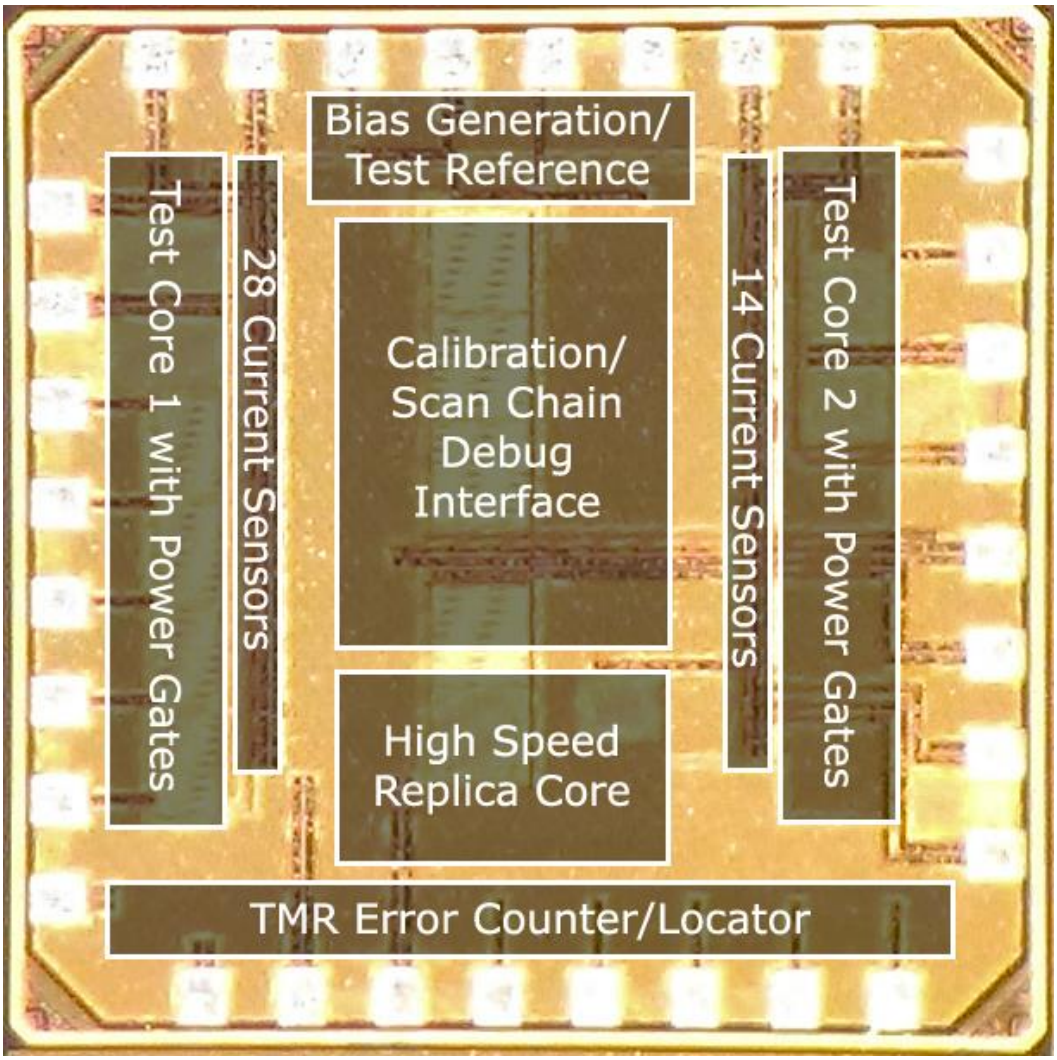


Figure 8.6: Die micro-photograph of test chip.

high-speed replica pipeline is identical to the two SIMD cores but designed to be operated at a higher voltage, guaranteeing faster operation. The output of each pipeline stage of each pipeline and the output of each current sensor is connected to the TMR block. If one of the SIMD lanes has an error relative to the replica core the TMR block will check to make sure the corresponding current sensor flags the error. The TMR block can track up to 8 errors per pipeline stage before needing to be scanned out.

8.4 Chip Measurement Results

In order to measure the operation of the current sensing test chip, a custom PCB was designed as well as a fully-automated software test platform (Fig. 8.7). The PCB was designed with several individually controllable low-noise, low-dropout voltage regulators. This allows for any supply voltage of the chip to be tuned from 1.2V down to a sub-threshold voltage of 0.2V. The PCB is also equipped with a dual-output 16-bit voltage DAC connected to the two analog sensor test inputs. All low-speed digital I/O signals go through level shifting before connecting with a USB controlled Ni-DAQ controller. Two 50-ohm high-speed clock signals are terminated next to the chip which are driven by an off-chip clock generator with programmable frequency, duty-cycle and skew.

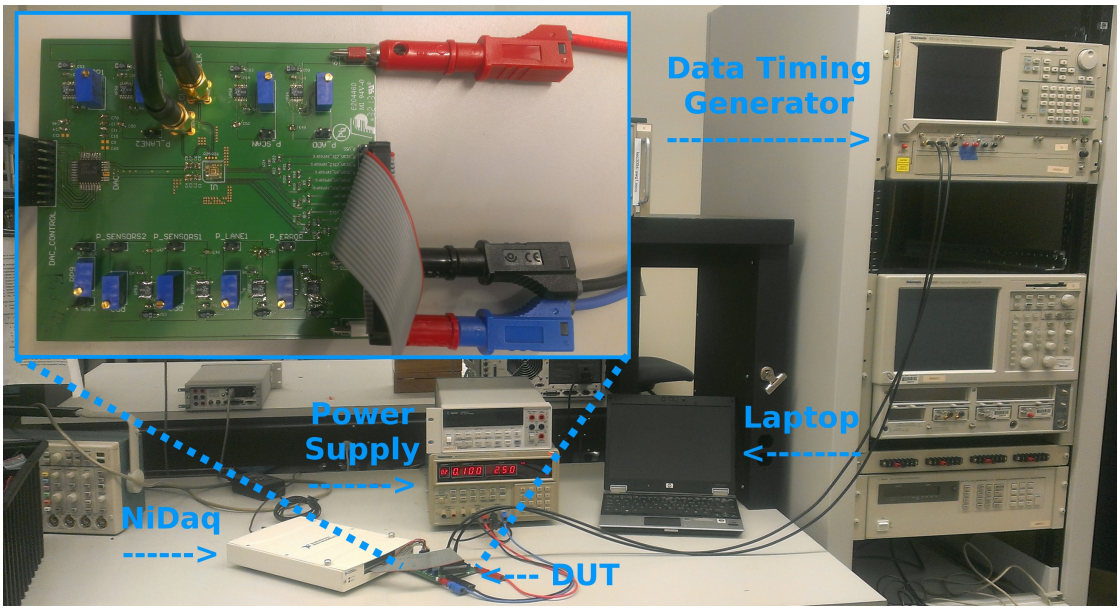


Figure 8.7: Lab test setup with custom designed test board and automated software running in LabView CVI.

8.4.1 Sensor Calibration Analysis

Not only is it critical that the sensor's offset be calibrated below the noise floor of its inputs but it must also be able to be tuned within several millivolts of the desired sensing margin. In order to ensure this critical part of the sensor works properly in this test chip the offset calibration was over-designed. The sensor was originally designed with 9 bits of calibration. The LSB was too small to work effectively across process variations so the number of effective bits were dropped to 8.

At 1V, 8 bits of calibration yields sub-1mV calibration steps which is more than what is required. As shown in Fig. 8.8 the worst-case offset for all 42 sensors

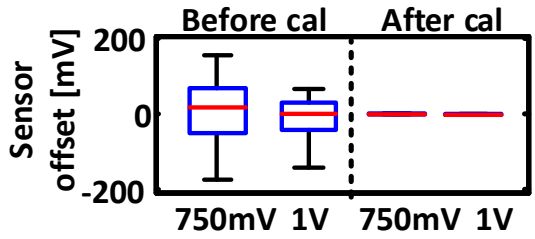


Figure 8.8: Measured comparator offset before and after calibration.

on the chip was less than +/-175 mV. By increasing bias current and achieving more than $1mV$ steps only 6 bits are required for proper tuning.

At 0.75V supply operation, the offset per sensor does go up, requiring one more bit of calibration. This makes the total required at 0.75V to be 7 bits, still lower than the available 8. As can be seen in Fig. 8.8 the offset of all sensors on the chip can be calibrated beyond the required offset with at most 7 bits.

8.4.2 Droop Plot Measurements

The sensors themselves can be used as on-die oscilloscopes to measure virtual supply droop in-situ. Because of their high precision, with sub-$1mV$ voltage accuracy and sub-$1ps$ timing accuracy from an off-chip source a great deal of information can be gathered. Fig. 8.9 shows four captured on-die voltage droop profiles, using the sensors as on-die oscilloscopes. In this particular computation (and many others), the adder is the critical path. Not only does this measurement prove that

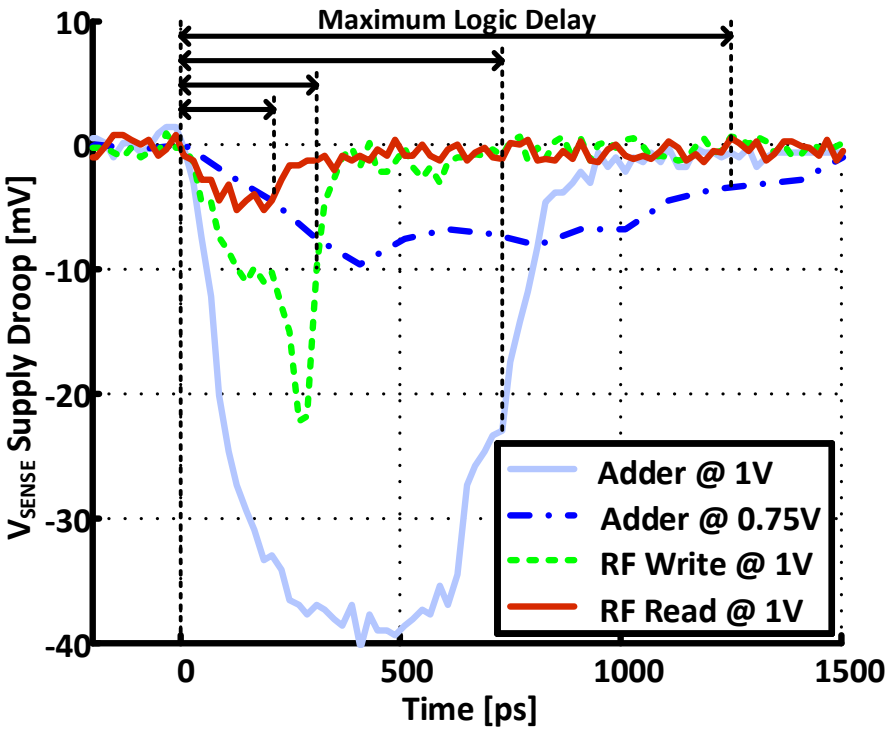


Figure 8.9: On-die voltage droop plots captured using the analog sensors as on-die oscilloscopes, for logic operating at 1.0V and 0.75V. Arrows at top denote when logic operation has finished.

the sensors are working, but it can also be used to determine hot-spots in computations and ensure designs are being partitioned correctly in the place-and-route phase. In this case it can be seen that better partitioning could be done to better balance the register file current consumption per sensor relative to the adder.

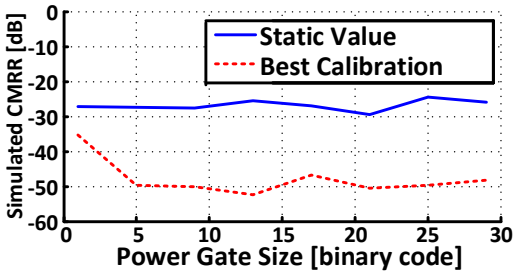


Figure 8.10: Simulated supply noise rejection at inputs of comparator after noise filtering.

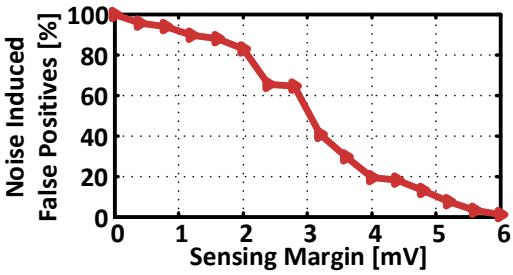


Figure 8.11: Sensing margin impact on noise-induced false positives at 1V supply.

8.4.3 Noise Analysis

Noise rejection can dramatically degrade the sensor performance, and is one of the limiting factors for robust analog-based error detection. In order to reject asymmetric supply noise on the sensor inputs, a small replica power gate is connected between the main digital supply and the non-inverting input of the comparator. This digitally-tunable replica power gate improves input noise rejection by 25dB over an analog sensor with no replica (Fig. 8.10).

The experimental measurements in Fig. 8.12 show that the noise floor can be improved to less than 7mV for a 1V supply, relaxing the requirements for the power gate size. The probability of a noise-induced false positive is therefore directly related to the power gate size and sensing margin set by the comparator. This probability of noise-induced false positives is shown in Fig. 8.11. The probability of a false positive does increase quickly. As the sensing margin decreases it is also important to note that it has no affect on false negatives. In most systems a small

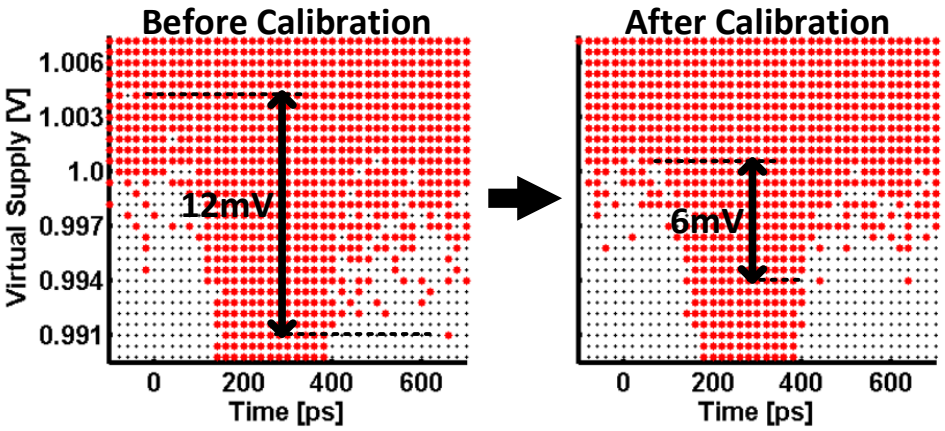


Figure 8.12: Measured on-die scatter plots of sensing droops before and after basic calibration of noise filtering.

amount of false positives can be tolerated but more often than not, false negatives can have catastrophic consequences.

8.4.4 Power Gate Sizing

Smaller power gate sizes increase the voltage droop margin while slightly penalizing the speed (Fig. 8.13). For a 1V supply, the power gate size requirement results in a 0.03% speed reduction. At 0.75V the power gate is programmed smaller due to less dynamic current, resulting in a 0.3% speed reduction. The chip could not be measured for VDD $<$ 0.75V since the minimum power gate size was too large, reducing the voltage droop amplitude below the sensor sensitivity. However, the sensors functioned properly below 0.55V.

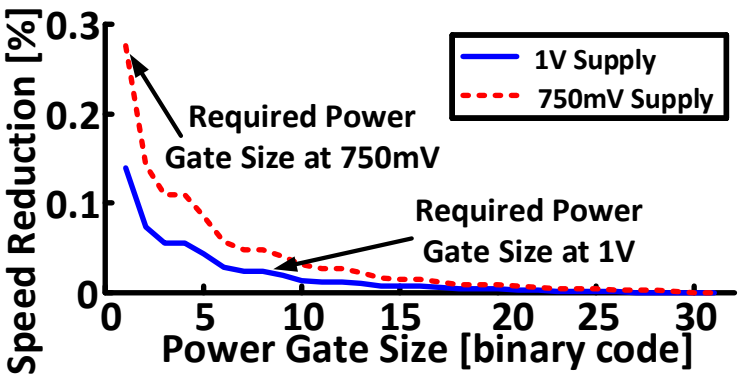


Figure 8.13: Power gate sizing for required voltage droop vs. impact on speed reduction.

8.4.5 Throughput Analysis

Fig. 8.14 shows the throughput and energy improvement of a SIMD lane using current sensing, compared with simulation models of TRC and Razor. It is clear that all three designs allow for a speedup beyond the margin set by the electronic design automation (EDA) tool. From this graph it appears that there is little difference between the three sensing methods as far as energy and throughout.

To better understand the comparative performance of each error detection technique throughput improvement (in percent) should be examined. Fig. 8.15 shows the throughput increase in percent above the EDA margin for all three designs. The CSCD method performs similar to Razor at 1V and outperforms both methods at 750mV. With an even smaller power gate size, the proposed CSCD is further expected to outperform when VDD $<$ 0.75V, since the widely varying delays are difficult to predict with the conventional methods.

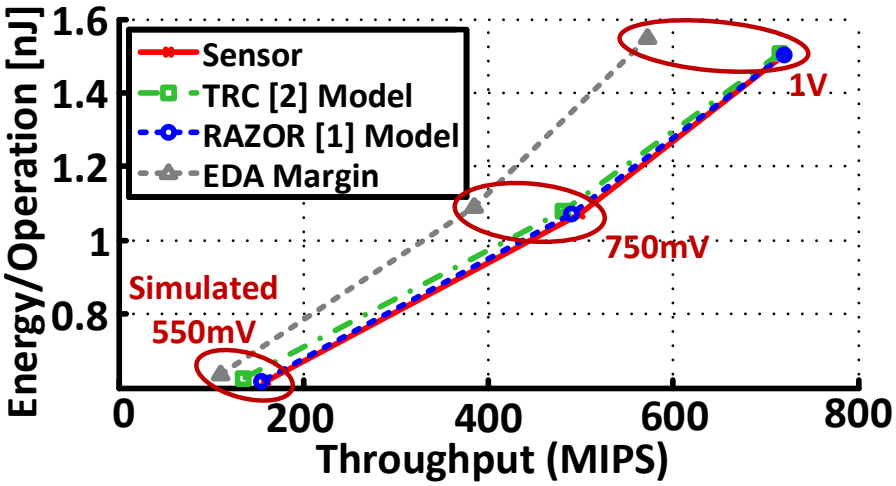


Figure 8.14: Comparison of throughput vs. energy efficiency at various supply voltages vs. conventional.

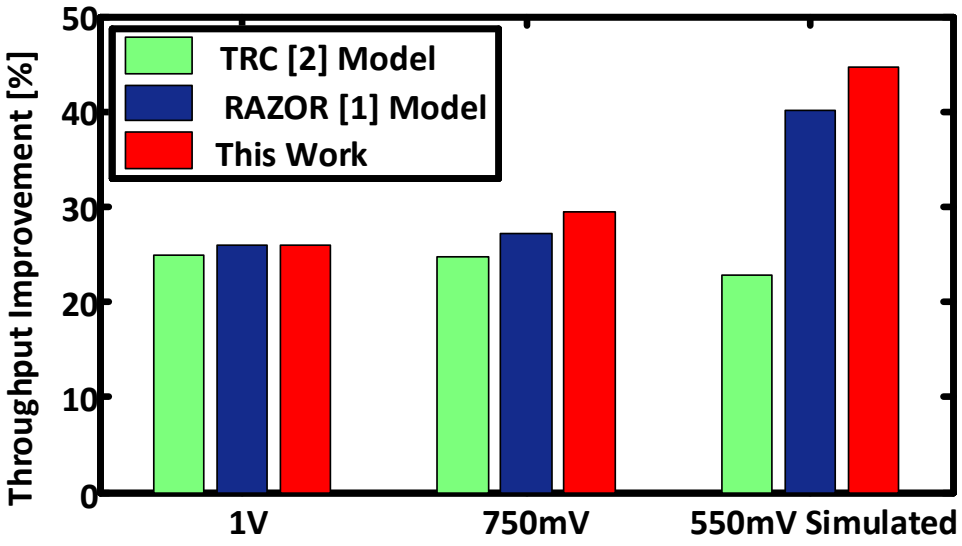


Figure 8.15: Throughput improvement beyond the EDA margin vs. conventional.

By inspection of Fig. 8.14 a slight decrease in energy/operation can be seen as throughput is improved at each voltage. This improvement is greater than 2% on average, because the complexity of the logic in the lanes being tested is not complex enough to overcome the larger percentage of flip-flop energy in the design. In larger designs a larger energy improvement is expected.

8.4.6 Results Summary

Table 8.2 shows a summary of the area and power measurement results for the test-chip presented in this chapter. As discussed in Section 8.4.4, because the sensors did not work below 0.75V due to the power gate size settings, only 1V and 0.75V are reported.

Table 8.2: Design summary of test chip.

	1V Supply	**0.75V Supply**
Technology	$65nm$ CMOS	
Die Area	$1mm$ x $1mm$	
Transistor Count	48k/Lane, 100/Sensor, 343k/Chip	
Area Overhead / Sensor	1.4% (8-bit cal), 0.6% (6-bit cal)	
Sensor Area Overhead / Lane	19-39% (8-bit cal), 8-17% (6-bit cal)	
Power / Lane	9.34 mW	4.95 mW
Power / Sensor	169 μW (225 fJ)	45 μW (69.2 fJ)

8.5 Future Work

This successful test chip design has proven that it is possible to implement current sensing at a high level of integration. It has also shown that there are possibilities for improvement to both the sensor design and system design to make CSCD more attractive.

8.5.1 Sensor Improvements

As mentioned in Section 8.4.1 the number of calibration bits were larger than needed and the LSB provided more granularity than what was required. In future sensor versions, the calibration DAC could be redesigned to have a similar dynamic range with fewer bits, dramatically reducing the area overhead.

In order to target the lower operating voltages of near-threshold, the dynamic range of the power gate sizing needs to be redesigned. As discussed in Section 8.4.4 the tune-able range of the power gate worked well at 1V and on down to 0.75V. However, in order push the supply voltage lower, the LSB of the power gate needs to be smaller to have a large enough sensing margin. Furthermore, it would be worth while to investigate non-linear power gate tuning as the response reported in Fig. 8.13 is clearly non-linear at smaller sizes.

8.5.2 System Improvements

For this chip, 14 sensors per core and 28 sensors per core are conservatively utilized, making the area/power overheads quite large (˜25%). If a system can be designed that guarantees timing errors only happen at the N closest gates to the flip-flop, some sensors can be removed. For example, imagine a data-path that contains 10 FO4 delays (or a chain of 10 inverters for explanation simplicity). If a maximum speedup of 20% is desired only the last two inverters need to be sensed. The overhead of this technique will scale smaller as longer chains are used.

On top of sensing only the end of logic chains, it may be possible to carefully design a system where it is only necessary to sense N critical paths. This would further reduce the overhead of this method and would be dependent on the maximum speedup desired. Accumulating these methods into a less comprehensive approach the overhead can be reduced to $< 2\%$, as less sensors are required to ensure proper operation.

8.6 Conclusions

In this chapter, an analog-based technique to detect timing errors in digital circuits operating at low supply voltages were presented. The test chip design and measurement results have shown that this method excels over existing digital methods due to the absence of a detection window and the ability to detect delays faster than the worst case, improving timing speculation and therefore performance. The sensors themselves are highly calibrateable allowing for the reliability concern to be

pushed to the sensors and out of the digital circuits they are protecting. A 65nm-CMOS test-chip is measured at 1.0V-0.75V, showing improvements in throughput and energy efficiency over traditional margining by 25% and 2%, respectively, with area/power overheads comparable to conventional digital detectors.

Chapter 9: Using Current Sensors to Detect Errors in a Radiation Environment

9.1 Introduction

In this chapter a *universal* SET (Single Event Transient) detection approach based on the CSCD sensors demonstrated in Section 8.3.2 is analyzed. This detector is *complete* in that it detects all possible SETs, and *efficient* because it leverages standard components that are compatible with current manufacturing processes. This detector relies on the realization that for any CMOS transition that occurs (erroneous or otherwise), current must flow through that affected circuit. If an SET occurs and changes the output of a circuit, transient current must be flowing at the same time when the circuit is expected to be stable (i.e. at the end of the clock period). The CSCD sensor is specifically designed for logic SETs and complements techniques for protecting memories and latches. The downside of this detector is that while SDC is eliminated, false alarms are introduced. This chapter includes a detailed discussion of the potential for false alarms and their impact on both processor and system performance. To the best of my knowledge, this is the first time that current sensing for SET detection within logic is described and

evaluated in detail, although it has been mentioned briefly in some prior work.

The rest of the chapter is organized as follows. First, the related work is discussed in Section 9.2. Then, the proposed detector is described in detail and a discussion is made on various considerations of its design and of potential false alarm in Section 9.3. After describing the detector, its incorporation in a processor is presented in Section 9.4. Section 9.5 then explains the evaluation methodology and results are presented in Section 9.6. The chapter is then concluded in Section 9.7.

9.2 Related Work

Previous approaches for detecting SETs in general combinational circuits have been based on some form of replication. Table 9.2 summarizes conventional techniques and compares them to the proposed approach. The most common and general approach is to replicate the entire circuit, as in standard *Dual-* and *Triple Modular Redundancy* (DMR or TMR) [123]. Alternatively, in specific cases of well-defined combinational logic circuits such as ALUs, only a simplified version of the circuit is added as a form of reduced replication. One well-known example of this technique is the use of arithmetic codes for error detection [124]. Arithmetic coding is limited to ALUs only and requires careful design and use. Another form of partial replication is *parity prediction* [125–127], which can be used with arbitrary circuits. However, the overhead of general parity prediction can be close to that of DMR [125, 127], and it is often selectively applied to mostly protect latches rather than logic, in which case overhead can be reduced but the potential for undetected

errors exists [126, 128, 129]. All replication techniques are costly in terms of energy and area, especially full replication.

An alternative group of approaches replicates in time rather than in space. Examples include running the same program, instruction, or operation multiple times [130–133]. Because SETs are rare events caused by random particle strikes, repeating computation on the same hardware can be used to provide redundancy. While this eliminates the requirement for redundant hardware, time replication negatively impacts performance and efficiency. Furthermore, if care is not taken to sufficiently separate the multiple execution instances in time, a long duration of error events resulting from a high-energy particle strike cannot not be detected.

Table 9.1: Comparison of particle-strike induced error detection mechanisms for combinational logic.

	DMR (space)	**DMR (time)**	**Coding**	**Razor**	**BICS**	**This work**
Universal?	Y	Y	N	Y	Y	Y
False negatives?	N	N[a]	Y[b]	Y	N	N
False positives?	N	N	Y	N	Y[c]	Y
Area overhead estimate	100%	$50-100\%$	$5-40\%$	$\sim 20\%$	$15-30\%$	$\sim 10\%$
Energy overhead estimate	100%	$50-100\%$	$5-40\%$	$\sim 20\%$	1%	$< 5\%$
Evaluated for logic?	Y	Y	Y	Y	N	Y
Easy to integrate?	Y	N	N	N	N[d]	Y
Main limitation	Overhead	False Negatives, complexity	ALU only, complexity	Overhead, min-delay, false negatives	Latchup renders inoperable	False positives

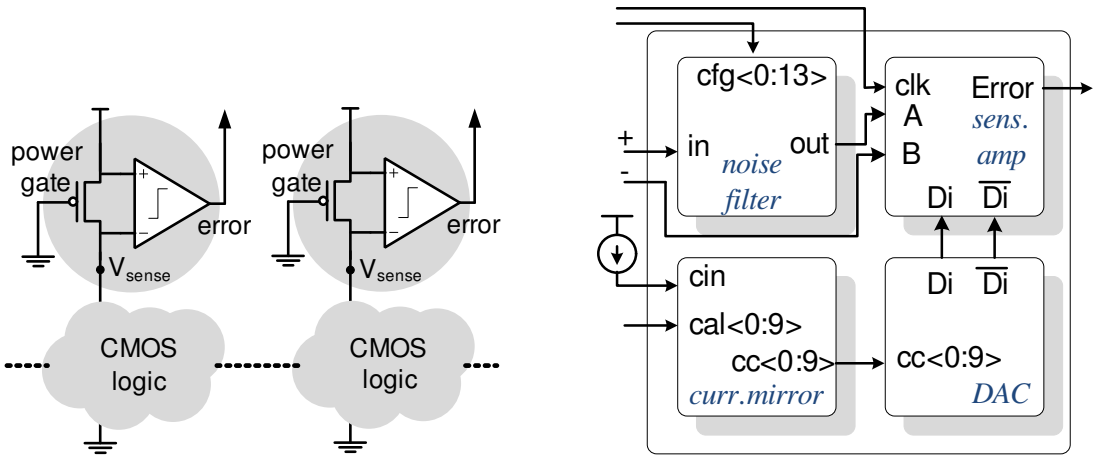
[a] Only if re-execution is delayed
[b] Can trade-off completeness with overhead.
[c] Not mentioned or evaluated.
[d] Latchup concerns.

A different approach for SET detection in combinational logic uses timing-speculation circuits, such as *Razor* [134], which was previously shown to provide some detection capability of particle strike induced errors [135]. The Razor flip-flop, which replicates the storage element, is used to compare the value latched by a pipeline register with the output of the previous logic stage slightly shifted in time with a small delay after it was clocked. A difference between these two registers indicates that an error has occurred. Unfortunately, for this technique to operate correctly, strict timing constraints are placed on the delays of the combinational logic, such as the bounding of the min-path delay and the maximum detection window, thereby increasing the area/energy overhead [136]. Furthermore, similar to temporal re-execution, this Razor technique may not be able to detect errors caused by extremely energetic particle strikes. Razor circuits work very well to detect timing related errors but may not operate correctly in the presence of SETs. The Razor circuit's ability to properly detect all SETs is dependent on both the operating frequency of the system as well as the length of the particle strike. For example if an SET occurs before the shadow latch and continues through the end of the clock cycle there is a small probability that Razor will not detect the error resulting in a false negative. A similar approach to using Razor is possible with some hardened latch designs, which also rely on two storage elements to achieve robustness [137].

9.2.1 Current Sensors

Current sensing completion/transition detection was introduced as a technique to design self-timed asynchronous logic circuits [138] by monitoring the current that flows through the supply to the on-chip logic. After the current profile of the on-chip logic has settled, the current dissipation will converge towards a steady-state leakage current, signifying that computation has finished. The idea of completion detection is similar to our SET detector, although the motivation and analysis is vastly different. Unfortunately, the mechanism presented in [138] is relatively complicated and is not amenable to CMOS integration.

Current sensing has been previously evaluated as a mechanism for detecting SETs in on-chip SRAM [139–141]. For these *Built-In Current Sensors* (BICS), an analog current sensor is placed between the bulk node of each transistor and its respective supply. BICS have been shown to work well for detecting SEUs (Single Event Upsets) in SRAM. Neto et al. [140] mentions briefly that BICS can be used to detect logic SETs, but unfortunately, because the sensor is connected to the bulk contact, there is a high possibility for latchup [142]. Latchup is an event that can occur in any CMOS process, where a resistance is formed between the power supply of a transistor and its bulk contact. In the case of BICS, resistors are deliberately connected between the power supply and the base of parasitic transistors in the design. An SET current pulse is likely to cause a latchup in this design, causing the BICS circuit to remain in this self-sustaining latchup state with a low resistance path between V_{dd} and GND. This V_{dd}-GND short will likely



(a) Sensor, logic, and power gate.

(b) Sensor components.

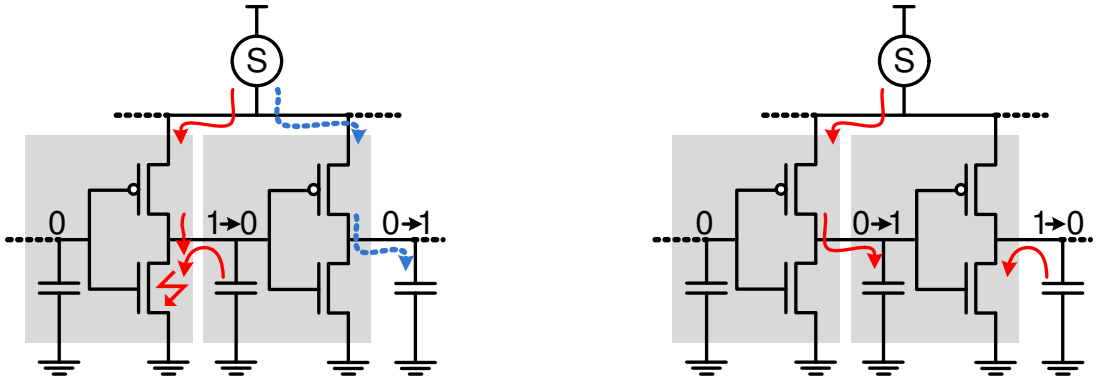
Figure 9.1: Block diagram of current sensor for SET detection.

lower the supply voltage of the rest of the connected logic causing every circuit connected to that supply to fail. Once latchup has occurred, the only way to remove it is to completely shutdown power. Latchup in the BICS configuration is highly likely to occur when an SET occurs, significantly degrading the effectiveness of this technique.

An alternative, single-supply current sensor design, which does not suffer from the problems mentioned above, was proposed in Chapter 8. This design is simpler than the previous approaches described above and utilizes a common power-gate sleep transistor as the voltage sensor (Fig. 9.1). This single-supply approach does not add any resistances between the bulk contact and supply, and is therefore not vulnerable to latchup. Further, the digital-assisted design, which consists of a sense-amplifier based comparator with a current-steering digital to analog converter (DAC) for offset calibration, is very power efficient and is designed for

modern CMOS VLSI process technology. For a $1.2V$ supply voltage in $90nm$ CMOS, the sensor requires $54fJ$/comparison and can complete a detection in less than $100ps$. The sensor is implemented with roughly 100 transistors for a robust and tunable design. This all-digital design scales well with technology, and we expect it to match the scaling of combinational logic circuits when scaling to smaller technology nodes, such as $22nm$ and $14nm$.

This technique correctly detects SETs because the current is sensed only at the clock edge. At that instant, no dynamic current should be flowing and the outputs should be stable, unless there is an undesired current induced by an SET. While some prior publications allude to the possibility of using current sensing to detect SETs, to the best of my knowledge, I provide the first detailed description and evaluation. Further, unlike previous work using current-based SET detection, this Chapter provides a detailed discussion of false alarms and their implication, which is a critical tradeoff in the overall system design and use.



(a) Stage 1 : strike and flip

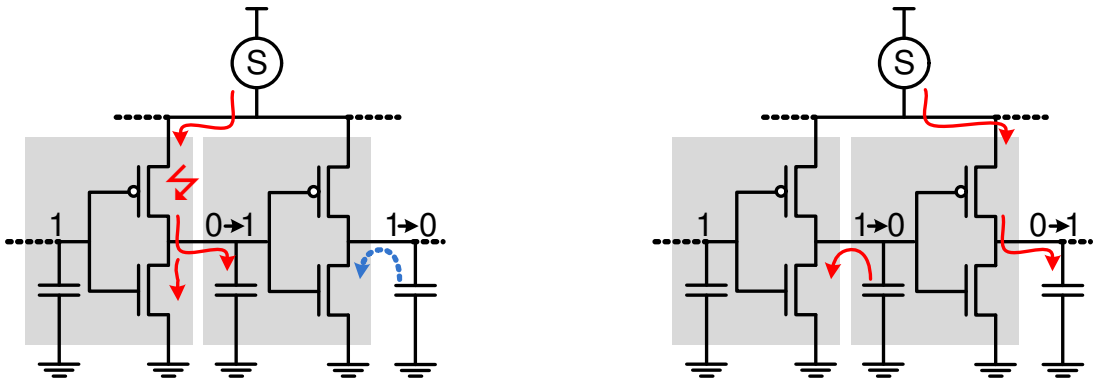
(b) Stage 2 : revert

Figure 9.2: SET strike at a closed NMOS transistor.

9.3 Current-Sensor SET Error Detector

9.3.1 SET Detector Operating Principles

Figs. 9.2–9.3 depict the particle strike scenarios for NMOS and PMOS transistors, respectively. For simplicity, without any loss of generality, a circuit consisting of two chained inverters is used, with the first inverter being struck. The first stage, in response to a particle strike, exhibits a large transient current flow from the generation of electron-hole pairs, which are then collected by the source and drain. If the current is large enough, it triggers a transition in the second inverter and potentially an SET. The second stage of the circuit reverts to steady state after the current pulse dissipates, and the correct inputs are the re-propagated. Note that for an SET to occur, the pulse must propagate to and be latched by the output flip-flop of the combinational logic circuit. Pulses that do not generate an SET are



(a) Stage 1 : strike and flip

(b) Stage 2 : revert

Figure 9.3: SET strike at closed PMOS transistor.

masked [143].

As shown in Figs. 9.2–9.3, there are two reasons for current to flow: the collected electrons and holes freed by the strike (solid red arrows); and the charge/discharge of the affected load gate as it first flips and then reverts (dashed blue arrows). For a NMOS strike, the SET current flows from V_{dd} and from the internal node capacitor to ground through the struck transistor. Electrons freed by the strike move towards V_{dd} and pass through the power gate by which the sensor can detect the current. As the internal node capacitor discharges, the load gate opens and additional current flows from V_{dd} until this "flipped" state stabilizes. In the revert stage, current is drawn from V_{dd} to bring the internal node back to a logical '1'.

In the strike stage for a PMOS transistor, current flows from V_{dd}, similar to the NMOS strike. This current flows both through the NMOS of the struck gate to ground and also charges the internal node capacitor until the state is "flipped" from a '0' to a '1'. However, unlike an NMOS strike, current does not flow through the load gate in the strike stage of a PMOS. Hence, the revert phase is essentially identical to an NMOS strike. Note that detecting this current is sufficient to indicate a potential SET, eliminating the need for a second current sensor on the V_{ss} line [138]. This insight leads to the adoption of a single-supply current sensor shown in Fig. 9.1. Because only one sensor is needed for a group of logic, and because it is only clocked once per cycle, it is an excellent candidate for SET detection and provides both efficiency and high coverage.

9.3.2 Masking and False Alarms

Not all SETs lead to an error that impacts the current computation. This proposed detector may wrongly identify such *masked* SETs as actual errors, causing a *false alarm*. Note that the proposed detector does not exhibit any false negatives and triggers on any possible SET.

There are five mechanisms by which an SET can be masked and therefore lead to a false alarm:

Electrical Masking. As the strike current pulse propagates through the circuit, it is attenuated by each affected transistor because of the electrical characteristics of the transistors (transistors with larger drive currents attenuate the pulse more), such that it does not cause an erroneous transition at the output. With any current detector, SETs that are masked by this mechanism may trigger a false alarm because current still flows through the gate transistor and sensor.

Temporal Masking. Because latches have a finite latching window near the end of a cycle, many SETs may already revert to the correct circuit state before being latched or not have time to propagate to the output before the latching window closes. Any current-sensing detector monitors the current at any point in the circuit within the latching window, indicating a potential SET and triggering an alarm. There are two reasons why a false alarm due to temporal masking may occur: (1) a transistor is in the strike stage of an SET and current is flowing as a direct result of the freed charge being collected; or (2) a transistor is in the revert stage and current is drawn to re-stabilize the circuit.

Logical Masking. Some SETs are masked because the output of the circuit does not depend on the internal value that was flipped. With this detector, even an SET that is logically masked may trigger a notification. Because temporal masking depends on the duration of the revert stage, logical masking does reduce the false alarm rate.

Architectural Masking. Modern processors often process instructions that never commit their values and use circuits that only impact performance rather than correctness. With an SET detector, it is up to the architecture to continue masking detected SETs, such as deferring taking action on a detected error until the instruction that it impacted is retired.

Application Masking. Some erroneous values can either be ignored or tolerated at the application level, which then decides whether to mask a detected error or not.

9.3.3 Sensor Calibration

The sensor must be sensitive enough to detect the small currents resulting from a particle strike. Achieving this sensitivity in the presence of aging, process, and temperature variations requires that the sensors be calibrated with respect to the static circuit currents. Thus, the sensor operates by tuning once at startup using a simple binary search offset calibration. For an eight-bit calibration, the sensor requires 9 clock cycles to calibrate. In order to provide robustness across varying temperatures, the sensor must be periodically recalibrated. A 5 degree C change

in temperature will change the sensor's offset by $1mV$, or approximately one calibration. Based on [144] the temperature is not expected to change more than 1 degree per minute, thus requiring a worst case recalibration interval of 5 minutes per sensor. Calibrating the sensor requires multiple iterations. A small number of iterations is typically sufficient because the drift is small over the period of time between recalibrations. Circuit simulations at 90nm indicate that each calibration iteration consumes roughly $450fJ$, including the energy of the calibration circuitry and sensor. While re-calibration is expected to complete within 10 cycles, initial calibration on startup may take > 100 iterations. Even this startup calibration, however, will complete within 100ns per sensor, assuming a 1GHz clock. The architectural aspects of recalibration and its overheads are discussed in the next section.

9.4 Architecture

The proposed SET detector can be applied to any CMOS combinational logic block and is an especially good fit within the pipeline of a modern out-of-order (OOO) processor. The mechanism is also an exceptionally good fit for complementing the error protection mechanisms of highly-reliable processors, such as the IBM Power 7 [129] or the Fujitsu SPARC64 [145]. To recap the key advantages of our technique, the detector protects any combinational logic with low overhead and detects the errors as they are being latched, such that the pipeline can be immediately stalled without propagating erroneous values. Alternative techniques

require redundant operations or logic to detect errors, and thus introduce delays while this mechanism operates very quickly.

9.4.1 SET Detection and Recovery

In the context of an OOO (out of Order) processor, the microarchitecture maintains its architecturally visible state separately from the in-flight state of instructions that are still in the pipeline. This provides a straightforward mechanism for isolating detected faults and errors. As long as the errors are detected before an in-flight instruction is committed to architectural state, the pipeline can be stalled and flushed, which reverts its state to a correct and consistent program-visible one. In essence, because this detector works within the pipeline, a detected SET immediately stalls the pipeline and initiates this flush-based recovery. No potentially-erroneous result is committed, and the pipeline restarts execution from the oldest correctly-committed instruction.

The approach above treats any potential SET as an actual error and ignores the possibility of architectural masking. An alternative approach is to locate the instruction, or instructions, that were impacted by the SET and *poison* them by marking them as potentially invalid. The commit stage of the pipeline will then be responsible for the pipeline flush. In this way, SETs that impact speculative instructions on an incorrectly predicted control path will not trigger an unnecessary flush. Given the expected rate of potential SETs, no significant difference in performance between these two approaches is anticipated. The impact of recovery

events, including those that follow false alarms, are evaluated in Section 9.6.3.

While the datapath and many control structures can be recovered with a pipeline flush, some components of the core and the processor cannot be protected in this way. Errors detected in the overall pipeline control, such as the commit unit itself for example, may not be recovered with a flush. Still, with our detector, such errors can be prevented from committing values to memory or I/O, and higher-level recovery, such as system-level checkpoint-restart [146], can restore execution state.

In this analysis, it is assumed that all combinational logic will be protected with the detector. Error signals will be aggregated within logically-connected components, e.g., a pipeline stage or a functional unit, to report errors. This follows a similar approach to that adopted by high-reliability architecture [128, 129]. We evaluate the impact on area and power of this full protection of logic. Overheads can be reduced by limiting protection in some cases. For example, some structures, such as a branch predictor, are used only to improve performance. An SET in the branch predictor is practically harmless because, at worse, it will lead to a temporary increase in branch misprediction rate and a transient dip in performance. It is also possible to leave some small structures unprotected and rely on the fact that the probability of a strike is low.

9.4.2 Interaction with Other Protection Mechanisms

SETs may impact all structures in the processor, not just combinational logic. In fact, SRAM arrays and latches are more vulnerable and must be protected as well for logic protection to make a difference. Large arrays are best protected with error checking and correcting (ECC) codes. Latches can be protected with parity-prediction [125–127], in which redundant logic is introduced to compute a parity bit, which is then latched along with the output of the pipeline stage. When the latches are read, a parity test is performed and a parity mismatch indicates an error. Note that this technique partially protects the logic as well. However, the cost of parity prediction is significant, and may equal that of DMR [125, 127]. The coverage of combinational logic SETs with parity prediction, may also be incomplete, especially when techniques are applied to reduce the overhead of the technique [126].

Another possibility for protecting latches is to used hardened latch designs [147–149]. Hardened latches significantly reduce the likelihood of an SET in a latch and can be simply applied by changing the cell library. While this is an appealing design methodology, it lacks the ability of parity prediction to partially protect logic. The combination of hardened latches and our detector offers a complete solution. In this way, each structure is protected with a separable mechanism, simplifying design and reducing the chance of undetected errors. We discuss the direct power and area overheads of the SET detectors in Section 9.6.4.

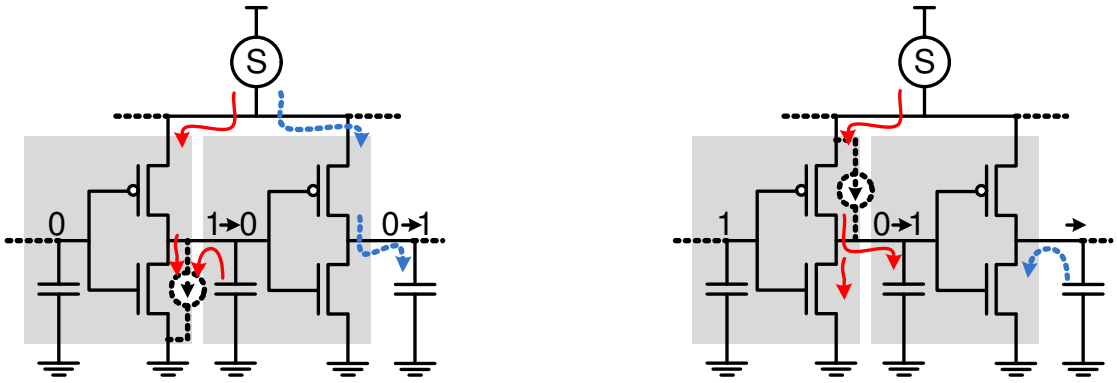
9.4.3 Calibration Architecture

A calibration circuit for a single sensor is roughly equivalent in area to the sensor. Calibration is only performed periodically, every several seconds, and therefore calibration circuits can be shared between multiple circuits. The area of the calibration unit is expected to grow roughly at O N where N is the number of sensors per calibration unit. Given this scaling, this design calls for a calibration circuit for every 1024 sensors, yielding a very low area overhead while still providing large calibration parallelism to keep recalibration fast and minimize wiring.

9.5 Methodology

9.5.1 Current Sensor Evaluation Methodology

After designing and testing the analog sensor for an IBM $90nm$ CMOS process, extensive spice simulations were conducted in HSPICE to evaluate it for detecting SETs. When simulating a circuit, particle strikes can be modeled by introducing a current source in parallel to the affected transistor between the transistor's source and drain (Fig. 9.4) to simulate the strike current pulse. The current pulse itself represents the physical behavior of the strike. Several models have been proposed for the pulse, including the popular single and double exponential pulse models [150, 151]. Because there is disagreement in the community about which pulse model to use and also uncertainty about the best values to use for the parameters of the models, a methodology has been developed specifically for evaluating the



(a) SET strike at NMOS (b) SET strike at PMOS

Figure 9.4: SET modeling in SPICE using a current source.

detector.

Instead of modeling the pulse based on characteristics of a strike, the single and double exponential models are approximated with a trapezoidal pulse with an empirically determined minimum pulse width that can cause an SET in a single FO1 inverter. If the detector correctly triggers when this minimum pulse is injected then it is guaranteed to detect any possible SET. Fig. 9.5 depicts this minimum pulse (in solid red) and also a much smaller pulse that does not cause an SET but can still trigger detection (in dashed green).

9.5.2 False Alarm Evaluation Methodology

To evaluate the false alarm rate for this detector, the SET rate estimation tool *BFIT* [152] was modified, for which the source code is readily accessible. BFIT accounts for all three circuit-level masking mechanisms: electrical, temporal, and

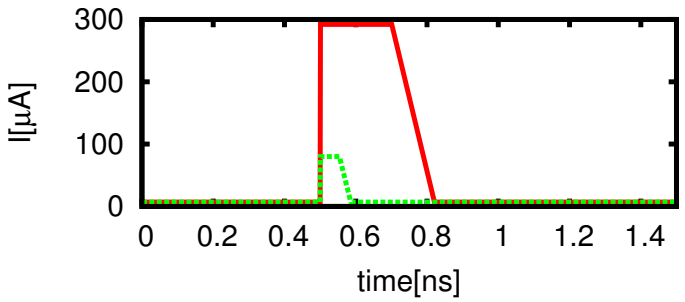


Figure 9.5: Minimum current pulse for an SET in an FO1 inverter (red); and a detectable pulse that is too small to generate an SET (green).

logical. The tool models the SET-related properties of the $45nm$ Nangate standard-cell library [153]. The model represents the impact of a particle strike as a function of the strike magnitude (collected charge), the time within the cycle that the strike occurred at, and the location of the cell within the input circuit. Unmodified BFIT was used to estimate the true SER (Soft Error Rate) for the set of ISCAS'89 benchmarks [154] included with the BFIT distribution.

To estimate the false-alarm rate, BFIT was modified to account for the interactions of the different masking effects and the current sensor as discussed in Section 9.3.2. To model the impact of reduced electrical masking, BFIT's model was extended to also count low-energy particle strikes as "SETs", when they occur near the end of a cycle. Specifically, a linear relation between strike time and strike magnitude was used to conservatively approximate the additional detector notifications. Also, the pulse-impact duration modeled in BFIT was extended by the duration of the revert stage to model the decreased temporal masking.

9.6 Results

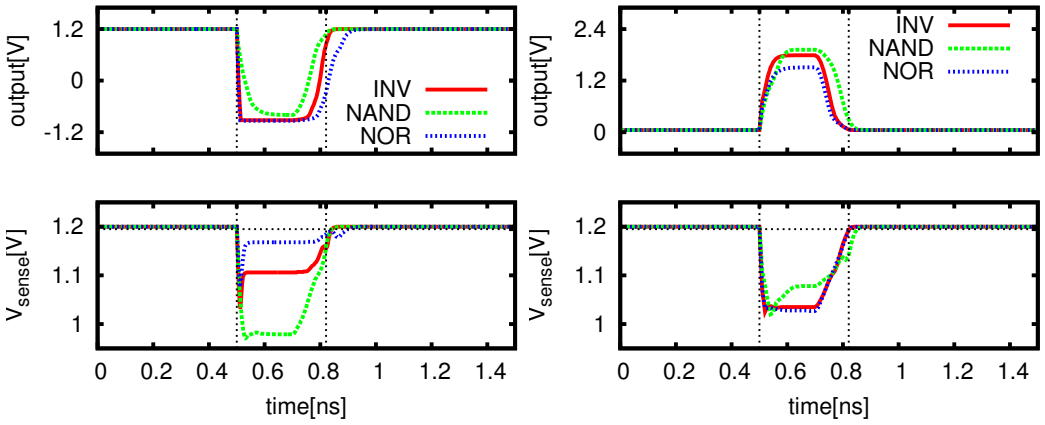
In this section, simulation results are presented while evaluating the proposed technique using the methodology described in Section 9.5. The discussion is broken into five subsections, focusing on the behavior of the detector with a single gate, detecting SETs in a small circuit, such as a 4-bit adder, analyzing the noise characteristics, evaluating the false alarm rate and the performance impact of recovery, and estimating the implementation overheads.

For each circuit-level simulation that evaluates the detector, a plot of V_{sense}, which represents the potential difference sensed at the power gate (Fig. 9.1) is shown. The V_{sense} trigger threshold is set to $1.195\,V$ (horizontal dotted line); a sensed value lower than the threshold indicates a potential SET. In addition, all the plots in this section contain vertical dotted lines that mark the beginning and the end of the current pulse (Fig. 9.5).

9.6.1 Single Gate Analysis

Fig. 9.6 shows the simulation results for the simplest logic building blocks – single gates. Each simulation is performed on a chain of 2 gates (2 INVs, 2 NANDs, or 2 NORs) such that the first gate is experiencing a particle strike and the second gate is serving as the first gate's load. The simulation shows that sampling V_{sense} at any point of time while the strike current persists will correctly identify the erroneous transition.

Fig. 9.7 shows the output and V_{sense} signals for an inverter in case of a low-



(a) Single gate - NMOS strike

(b) Single gate - PMOS strike

Figure 9.6: Single gate response to an SET-generating strike current pulse.

energy particle strike that generates a short pulse with a peak of only $80\mu A$. This current is not enough to flip the output of the first gate, but is large enough to trigger the detector (the current sensor drops below its threshold). In order to avoid potential false negative detections—SETs that are not detected can silently corrupt data—the detector is over-sensitive and exhibits false positive detections, or false alarms as discussed in Section 9.3.2.

9.6.2 4-bit Ripple-Carry Adder

To evaluate a more complex circuit, particle strikes were simulated within a 4-bit ripple-carry adder. In each simulation, the inputs to the adder were held steady and a single particle strike, minimum current pulse was introduced to a single transistor. We repeated the experiment five times, introducing the SET to five

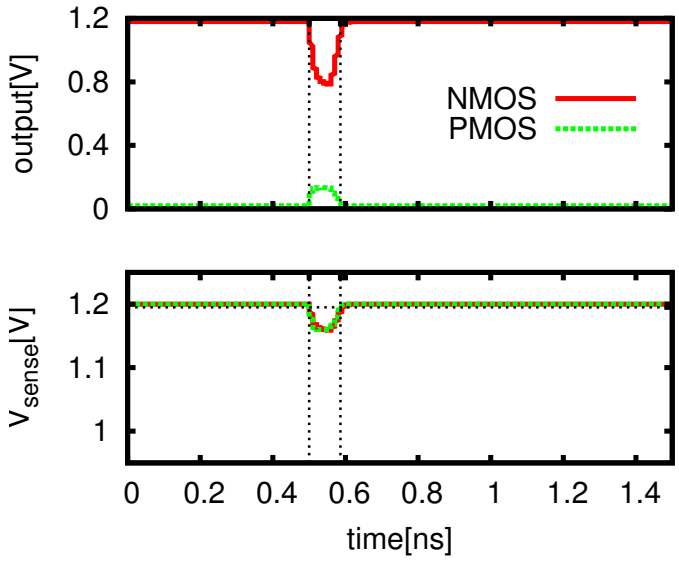


Figure 9.7: Single gate response to a low-energy particle strike current pulse. Top sub-figure is gate output and bottom sub-figure is the current sensor output.

distinct locations within the adder Fig. 9.8.

Fig. 9.9 shows the results for the five experiments, showing that any potential SET is detected. The top graph in each of the subfigures shows the strike current pulse, the middle graph shows the affected adder output, and the bottom graph shows the sensor output (V_{sense}). In all experiments, the SET propagated to the output and in all experiments the detector correctly indicates an error (V_{sense} lower than the threshold).

Fig. 9.10 shows the simulated adder *Cout* output and the sensor responses to a strike current over several cycles of operation. The figure demonstrates both the effectiveness of the detector as well as temporal latching. The clock is shown at the top of the figure and latching windows are indicated with grey boxes. In the

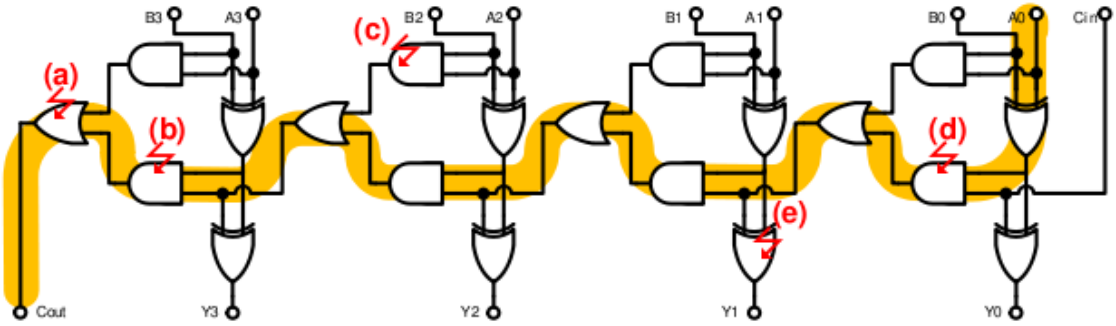


Figure 9.8: Five simulated SET scenarios for a 4-bit adder. Critical path is highlighted, and the 5 strike scenarios are marked a–e (strike e is to an inverter within the XOR).

first cycle, the inputs are held constant and no strike occurs. At time $t = 1ns$, the inputs are changed and both the output and sensor react and stabilize before the next latching window. The inputs are changed again at $t = 2ns$ and this is shortly followed by a strike current at $t = 2.1ns$ (the current pulse duration is indicated by the dotted lines). The pulse dissipates well before the next cycle and is temporally masked and correctly not detected. The inputs are changed again at $t = 3ns$ and another strike pulse is introduced at $t = 3.7ns$. This pulse leads to an SET as the output is still flipped while being latched. The detector correctly identifies this SET as V_{sense} is clearly below the threshold at the clock edge ($t = 4ns$). Again, it is important to note that the sensor only operates at the clock edge when the outputs should be stable and no current should be flowing (marked in grey in Fig. 9.10). As explained in Section 9.3, the sensor is fast enough to detect an error before it is propagated even if its operation is controlled by the same clock as the latch.

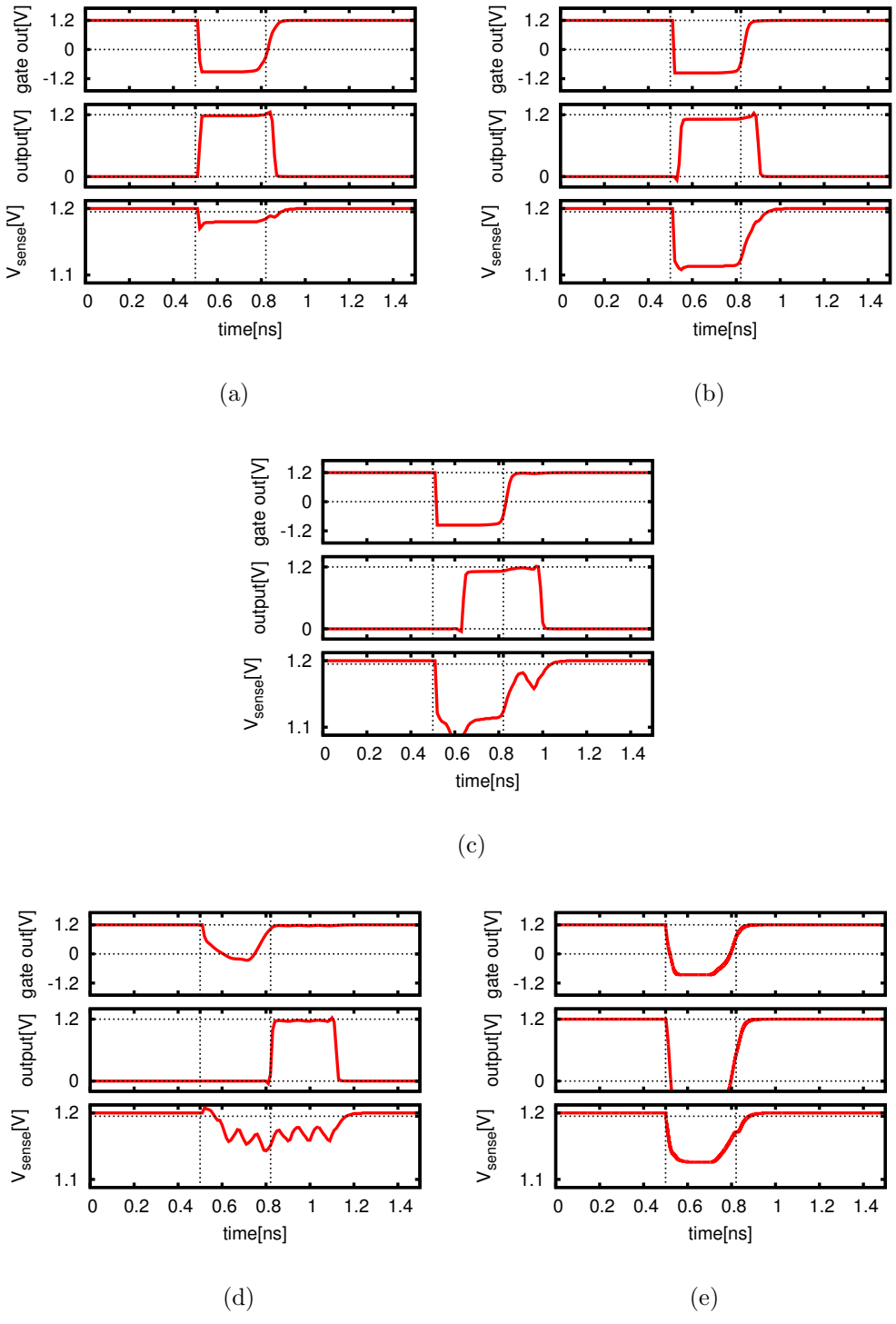


Figure 9.9: Adder response to five scenarios of SET-generating strike current pulses.

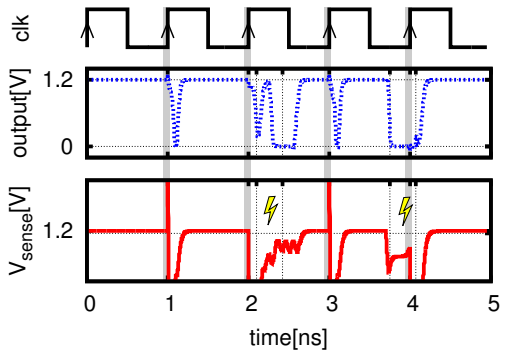


Figure 9.10: Three cycles of adder operation showing Cout and V_{sense}. Latching windows in grey, strike current duration within dotted lines. Current is sensed only within the grey latching window and normal currents are effectively masked.

9.6.3 False Alarm Rate Analysis and Performance Impact

While it has been shown that the proposed current sensor exhibits no false-negatives and triggers on all errors, this completeness comes at a cost of false alarms. We follow the methodology described in Section 9.5.2 and present results for the BFIT-supplied ISCAS'89 benchmark circuits in Fig. 9.11. The figure shows results for the seven circuits with both a $1GHz$ and a $2GHz$ clock frequency. Each bar represents the total detector notification rate (in units of FIT on the left axis, where 1 FIT is equal to 1 SET every 10^9 hours of operation. The lower part of each bar (shown in red) is the true unmasked SER, or true positive rate (TP) whereas the top part (shown in green) is the rate of false positives (FP). The figure also shows the *false alarm rate* (FAR) for each circuit (dotted line against the right axis), which is commonly defined as $FAR = \frac{FP}{FP+TP}$.

Three interesting observations about the results can be made. First, the false-alarm rate is fairly high at $87-99\%$ at both $1GHz$ and $2GHz$ operation. Second,

even with this fairly high rate, the absolute frequency of notifications are still extremely low at no more than 10FIT. Even for larger circuits (e.g., entire processors) it is expected that SERs in logic will not exceed 10^5 FIT, or roughly one error per year. Even at a false-alarm rate of 99%, this conservatively implies at most 2 false alarms per week per processor. Because a large fraction of all combinational logic circuits are within the OOO cores, a large fraction of false alarms and SER events lies within the cores as well. As explained in Section 9.4, such events can generally be isolated within the speculative state of the OOO core and recovered with a pipeline flush. A flush has a small overhead, which is measured in *ns*. Thus, two recovery events per week have zero impact on performance.

Even when projecting to a large system and assuming software must step to handle recovery, the expected performance overhead is still low. For example, considering a large-scale HPC machine, recent work has shown that even very high error rates can be tolerated without limiting scalability [155, 156]. Further more, it is possible for each processor to be recovered independently, and then even if the two events per week require relatively long recovery time, expected performance is barely impacted.

9.6.4 Implementation Overheads

Sensor Area and Power Overheads. In order to obtain the required sensitivity to detect SETs, the ratio of the dynamic-to-leakage current (or Ion/Ioff) of the combinational logic must be as small as possible. If high-leakage, faster

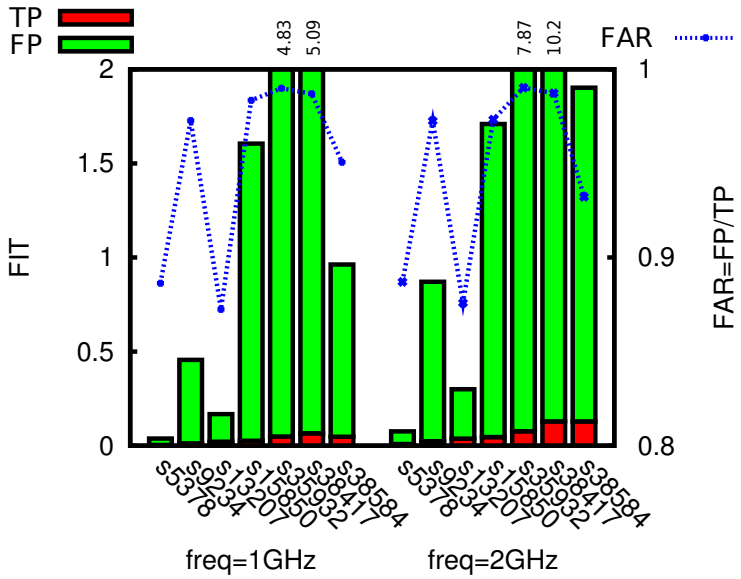


Figure 9.11: True and false positive FIT rates (lower and upper part of bars on left axis) and false-alarm rate (dotted line on right axis) for ISCAS'89 benchmark circuits. The benchmark names are along the x-axis.

transistors are used, such as those in low-V_{th} (LVT) cell libraries, more sensors must be employed for the same logic block. For example, HSPICE simulations in 90nm-CMOS show that a single sensor, which is composed of 100 transistors, can sense a maximum of approximately 200 LVT combinational logic transistors. Even this high overhead is still better than the generic protection techniques that rely on replication (see Section 9.2).

The efficiency of the detector can be improved by increasing the number of logic transistors it protects. This can be done without sacrificing accuracy in two ways. The first is by requiring a slightly lower operating frequency to enable the sensor to stabilize without the activity of the combinational logic transistor pulling the virtual supply too low. The second technique is to use transistors with better

Ion/Ioff ratios, such as those with high-V_{th} (HVT) or those that use Fin-FET and other non-planar devices [157].

With these two techniques, estimates indicate that a single 100-transistor sensor will be able to handle a combinational logic block of 1000 transistors while only increasing delay negligibly. With this sensor configuration, the power overhead of the sensors is less than 5%, when compared with the actual combinational logic. Each sensor requires only $500nW$ to operate, but is able to protect 1000 combinational logic transistors. Therefore, only 10^6 detectors are needed to protect the roughly 10^9 or so logic transistors in modern processors. While the processor logic consumes $>$ 10W or more, the detectors only require 0.5W. Area increase is estimated to be directly proportional to the number of transistors.

Calibration Overheads. With $1,024$ sensors sharing a single calibration unit, the entire set can be calibrated within roughly $1ms$. Such a short delay is not noticeable at boot time and provides ample scheduling freedom when considering the fact that recalibration need only occur every 5 seconds. Thus, calibration does not impact performance. With respect to power consumption, each recalibration is expected to complete in 10 or so iterations, with each iteration consuming 450fJ per iteration. Assuming a recalibration every 5s, this corresponds to a per-sensor recalibration current of 450fA, which is negligible when compared to the overall power of the sensor at $500nW$.

9.7 Conclusions

This chapter describes, evaluates, and discusses a new solution to detecting particle strike induced single-event upsets in combinational logic. By directly monitoring for currents that must flow if an SET occurs using a current sensor, the detector eliminates the need for high-overhead redundant logic while exhibiting complete detection coverage for any arbitrary combinational logic circuit. The sensor is coupled with a power-gating sleep transistor, such that this technique can be readily integrated with common design flows. Utilizing the sleep transistor as part of the detector also reduces its overhead, and the detector requires approximately only $500nW$ to operate (per logic block). The interactions between this technique and error masking mechanisms are analyzed and the expected false alarm rate for the detector is evaluated. While the false alarm rate is significant, its impact on actual processor behavior is minimal, because recovery events are still expected to be rare (< 2 times per week) and efficient recovery mechanisms are available.

The detector does increase die area and power consumption of combinational logic, but does so to a lesser degree than prior approaches for error detection. While the area of combinational logic blocks increases by about 10%, power consumption increase is less than 5%. The relatively high false alarm rate is thus the main downside of the proposed approach. However, the orders of magnitude improvement in power overhead combined with entirely eliminating the risk of silent data corruption from combinational logic is a favorable trade-off. With appropriate support from the architecture and relying on software resilience in rare cases,

the impact of the increased false alarms has a much smaller impact on overall performance and efficiency than with replication-based detection mechanisms. The calibration procedure may also have an impact on large parallel applications as was discussed at the end of the chapter.

Chapter 10: Conclusions

Near/Sub-threshold circuit designers are challenged with the task of making circuits reliable in the presence of variations beyond traditional requirements. The methods presented in this dissertation provide a large step forward in making near/sub-threshold digital circuit operation a viable operating point in the near future.

In Chapter 4 a design automation methodology was presented that aimed to lower the impact of variations on synthesized digital circuits. It was shown that large designs can benefit from a synthesis-based approach that removes nonperforming logic gates from the synthesizer as opposed to redesigning the entire library. A case study was carried out on this methodology in Chapter 5 where it was shown that a methodology such as this works best on large designs while requiring minimal design time.

Chapter 6 introduced two circuit-level methods for detecting errors in near/sub-threshold (TACD and CSCD) which were used to improve asynchronous micropipeline techniques. These methods where then adapted successfully for synchronous operation in Chapter 7 which gave them more value to modern microprocessors. CSCD, being of specific interest due to it's large speedup capabilities was demonstrated

in a $65nm$ test-chip in Chapter 8. The test-chip performed well and was shown to outperform the most recently proposed methods at near-threshold voltages.

The CSCD design was shown to have a wide range of uses, as it was shown in Chapter 9 to not only work with circuits that experience high levels of variations but also detect errors from radiation events. An in-depth analysis was done to show that even though the sensors generated a high number of false positive errors the relative performance impact was negligible. This opens the door for reliable radiation-tolerant digital logic as smaller process nodes make this a growing concern.

10.1 Final Thoughts

While energy efficiency will *always* be a major concern in the microprocessor industry, it is unknown exactly how it will evolve over the next ten years. With that being said, energy-efficient digital circuits have become a commodity in nearly every market segment, making it uncertain if reliability is of true concern for many applications. Smart-phone crashes and WiFi dis-connectivity plague society but don't seem to bother us as much as we might expect. However, of the applications that do need to reliably be energy-efficient (space applications, mission-critical servers, etc.), 99.999% perfection is expected.

In my opinion, all near/sub-threshold digital circuits of the future will be derived from asynchronous counterparts. Resiliency problems in general are often solved with asynchronous solutions today as is much of the work in this disserta-

tion. Often times designers do not necessarily realize that their designs are derived from asynchronous ideologies. I believe that taking a step back and identifying how a problem can be solved in an asynchronous manner can give valuable insight into what the best synchronous or pseudo-synchronous solution could be.

10.2 Future Work

Although this work presents viable methods for near/sub-threshold circuit resiliency, more work needs to be done before there can be widespread adoption of these methods. In the case of the proposed CSCD method it is necessary to have a well-understood, streamlined design that can be easily ported to future process nodes. This portability needs to come in the form of design tool integration and a solid foundation of trade-off analysis such as power, area, and detection capability.

In order to enable widespread adoption, CSCD needs further research. The design presented in this dissertation would value greatly from improvements to its circuit design. The designs presented in this dissertation were over-designed as a proof of concept. Much more work can be done to simplify the designs, improving the dynamic range while reducing the number of calibration bits required for correct operation – at the same time, lowing the area and power overheads. To that end, the overhead of shared calibration circuitry would benefit if it were to be reduced with further research.

Bibliography

[1] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: reclaiming moores law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[2] J. Kwong and A. Chandrakasan, "Variation-driven device sizing for minimum energy sub-threshold circuits," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, Oct. 2006, pp. 8–13.

[3] E. Krimer, R. Pawlowski, M. Erez, and P. Chiang, "Synctium: a near-threshold stream processor for energy-constrained parallel applications," *Computer Architecture Letters*, vol. 9, no. 1, pp. 21–24, Jan. 2010.

[4] J. Lorenz, E. Bär, T. Clees, P. Evanschitzky, R. Jancke, C. Kampen, U. Paschen, C. Salzig, and S. Selberherr, "Hierarchical simulation of process variations and their impact on circuits and systems: results," *Electron Devices, IEEE Transactions on*, pp. 1–8, 2011.

[5] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32b arm isa processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 284–285.

[6] K. Bowman, J. Tschanz, S. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. De, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 194–208, January 2011.

[7] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S.-L. Lu, T. Karnik, and V. De, "Energy-efficient and metastability-immune timing-error detection and recovery circuits for dynamic variation tolerance," in *Integrated Circuit Design and Technology and Tutorial, 2008. ICICDT 2008. IEEE International Conference on*, June 2008, pp. 155–158.

[8] K. Bowman, J. Tschanz, C. Wilkerson, S.-L. Lu, T. Karnik, V. De, and S. Borkar, "Circuit techniques for dynamic variation tolerance," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, Jul 2009, pp. 4–7.

[9] E. Mintarno, J. Skaf, R. Zheng, J. Velamala, Y. Cao, S. Boyd, R. Dutton, and S. Mitra, "Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 5, pp. 760–773, May 2011.

[10] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De, "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance," in *VLSI Circuits, 2009 Symposium on*, June 2009, pp. 112–113.

[11] K. Agarwal and S. Nassif, "Characterizing process variation in nanometer cmos," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 396 –399.

[12] A. Sedra and K. Smith, *Microelectronic circuits*, ser. Oxf Ser Elec Series. Oxford University Press, Incorporated, 2010. [Online]. Available: http://books.google.com/books?id=wDQJQQAACAAJ

[13] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 75–75.

[14] J. Benedetto, P. Eaton, D. Mavis, M. Gadlage, and T. Turflinger, "Digital single event transient trends with technology node scaling," *Nuclear Science, IEEE Transactions on*, vol. 53, no. 6, pp. 3462–3465, 2006.

[15] Y. Li, E. Cheng, S. Makar, and S. Mitra, "Self-repair of uncore components in robust system-on-chips: An opensparc t2 case study," in *Test Conference (ITC), 2013 IEEE International*, Sept 2013, pp. 1–10.

[16] A. Johnston, G. Swift, and D. Shaw, "Impact of cmos scaling on single-event hard errors in space systems," in *Low Power Electronics, 1995., IEEE Symposium on*, Oct 1995, pp. 88–89.

[17] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *Computers and Digital Techniques, IEE Proceedings* -, vol. 143, no. 5, pp. 301–307, September 1996.

[18] T. Liu and S.-L. Lu, "Performance improvement with circuit-level speculation," in *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd Annual IEEE/ACM International Symposium on*, 2000, pp. 348–355.

[19] L. Benini, E. Macii, M. Poncino, and G. De Micheli, "Telescopic units: a new paradigm for performance optimization of vlsi designs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 220–232, March 1998.

[20] Y.-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "An efficient mechanism for performance optimization of variable-latency designs," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 976–981.

[21] S. Hassoun and C. Ebeling, "Architectural retiming: pipelining latency-constrained circuits," in *Design Automation Conference Proceedings 1996, 33rd*, June 1996, pp. 708–713.

[22] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. Kim, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE MICRO*, pp. 10–20, 2004.

[23] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "Razor II: In situ error detection and correction for PVT and SER tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 32–48, January 2010.

[24] P. Franco and E. McCluskey, "On-line delay testing of digital circuits," in *VLSI Test Symposium, 1994. Proceedings., 12th IEEE*, Apr 1994, pp. 167–173.

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

[26] R. Sproull, I. Sutherland, and C. Molnar, "The counterflow pipeline processor architecture," *Design Test of Computers, IEEE*, vol. 11, no. 3, p. 48, 1994.

[27] Y. Tamir and M. Tremblay, "High-performance fault-tolerant vlsi systems using micro rollback," *Computers, IEEE Transactions on*, vol. 39, no. 4, pp. 548–554, Apr 1990.

[28] A. Drake, R. Senger, H. Singh, G. Carpenter, and N. James, "Dynamic measurement of critical-path timing," in *Integrated Circuit Design and Technology and Tutorial, 2008. ICICDT 2008. IEEE International Conference on*, June 2008, pp. 249 –252.

[29] K. Bowman, C. Tokunaga, J. Tschanz, A. Raychowdhury, M. Khellah, B. Geuskens, S.-L. Lu, P. Aseron, T. Karnik, and V. De, "Dynamic variation monitor for measuring the impact of voltage droops on microprocessor clock frequency," in *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, September 2010, pp. 1–4.

[30] D. Sorin, M. Martin, M. Hill, and D. Wood, "Fast checkpoint/recovery to support kilo-instruction speculation and hardware fault tolerance," *Dept. of Computer Sciences Technical Report CS-TR-2000-1420*, 2000.

[31] M. Prvulovic, Z. Zhang, and J. Torrellas, "Revive: cost-effective architectural support for rollback recovery in shared-memory multiprocessors," in *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on.* IEEE, 2002, pp. 111–122.

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

[33] T. R. N. Rao, *Error coding for arithmetic processors.* Orlando, FL, USA: Academic Press, Inc., 1974.

[34] J.-C. Lo, "Reliable floating-point arithmetic algorithms for error-coded operands," *Computers, IEEE Transactions on*, vol. 43, no. 4, pp. 400–412, Apr 1994.

[35] J.-C. Lo, S. Thanawastien, and T. Rao, "Concurrent error detection in arithmetic and logical operations using berger codes," in *Computer Arithmetic, 1989., Proceedings of 9th Symposium on*, September 1989, pp. 233–240.

[36] D. Strukov, "The area and latency tradeoffs of binary bit-parallel bch decoders for prospective nanoelectronic memories," in *Proc. Asilomar Conf. Signals Systems and Computers*, Oct 2006.

[37] D. M. Andrews, "Using executable assertions for testing and fault tolerance," in *9th Fault-Tolerance Computing Symposium*, 1979.

[38] A. Mahmood, D. J. Lu, and E. J. McCluskey, "Concurrent fault detection using a watchdog processor and assertions," in *1983 International Test Conference*, Oct 1983, pp. 622–628.

[39] M. Z. Rela, H. Madiera, and J. G. Silva, "Experimental evaluation of the fail-silent behavior in programs with consistency checks," in *FTCS'96: Proceedings of the Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing*, 1996, p. 394.

[40] J. M. Wozniak, A. Striegel, D. Salyers, and J. A. Izaguirre, "Gipse: Streamlining the management of simulation on the grid," in *ANSS'05: Proceedings of the 38th Annual Symposium on Simulation*, 2005, pp. 130–137.

[41] V. Balasubramanian and P. Banerjee, "Compiler-assisted synthesis of algorithm-based checking in multiprocessors," *Computers, IEEE Transactions on*, vol. 39, no. 4, pp. 436–446, Apr 1990.

[42] A. Al-Yamani, N. Oh, and E. McCluskey, "Performance evaluation of checksum-based abft," in *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on*, 2001, pp. 461–466.

[43] K.-H. Huang and J. Abraham, "Algorithm-based fault tolerance for matrix operations," *Computers, IEEE Transactions on*, vol. C-33, no. 6, pp. 518–528, June 1984.

[44] P. Banerjee, J. Rahmeh, C. Stunkel, V. Nair, K. Roy, V. Balasubramanian, and J. Abraham, "Algorithm-based fault tolerance on a hypercube multiprocessor," *Computers, IEEE Transactions on*, vol. 39, no. 9, pp. 1132–1145, September 1990.

[45] A. Reddy and P. Banerjee, "Algorithm-based fault detection for signal processing applications," *Computers, IEEE Transactions on*, vol. 39, no. 10, pp. 1304–1308, Oct 1990.

[46] J.-Y. Jou and J. Abraham, "Fault-tolerant fft networks," *Computers, IEEE Transactions on*, vol. 37, no. 5, pp. 548–561, May 1988.

[47] A. Mishra and P. Banerjee, "An algorithm-based error detection scheme for the multigrid method," *Computers, IEEE Transactions on*, vol. 52, no. 9, pp. 1089–1099, Sept. 2003.

[48] J. Wensley, M. Green, K. Levitt, and R. Shostak, "The design, analysis, and verification of the sift fault tolerant system," in *Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press, 1976, pp. 458–469.

[49] B. Nicolescu, R. Velazco, M. Sonza-Reorda, M. Rebaudengo, and M. Violante, "A software fault tolerance method for safety-critical systems: Effectiveness and drawbacks," in *Integrated Circuits and Systems Design, 2002. Proceedings. 15th Symposium on*. IEEE, 2002, pp. 101–106.

[50] N. Oh, S. Mitra, and E. McCluskey, "Ed4i: Error detection by diverse data and duplicated instructions," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 51, no. 2, 2002.

[51] G. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. August, "Swift: Software implemented fault tolerance," in *Proceedings of the international symposium on Code generation and optimization*. IEEE Computer Society, 2005, pp. 243–254.

[52] C. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 397–404, September 2005.

[53] R. Hamming, "Error correcting and error detecting dodes," *Bell System Technical Journal*, vol. 29, pp. 147–160, Apr 1950.

[54] M. Hsiao, "A class of optimal minimum odd-weight-column sec-ded codes," *IBM Journal of Research and Development*, vol. 29, pp. 147–160, Apr 1950.

[55] 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, March 1984.

[56] S. Lin and D. J. C. Jr., *Error control coding: fundamentals and applications.* Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1983.

[57] M. Turnquist, E. Laulainen, J. Makipaa, M. Pulkkinen, and L. Koskinen, "Measurement of a timing error detection latch capable of sub-threshold operation," in *NORCHIP, 2009*, Nov. 2009, pp. 1–4.

[58] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, *Sub-threshold design for ultra low-power systems*, ser. Integrated Circuits and Systems. Springer, 2006.

[59] A. Wang and A. Chandrakasan, "A 180mv fft processor using subthreshold circuit techniques," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, 2004, pp. 292–529 Vol.1.

[60] S. C. Jocke, J. Bolus, S. Wooters, A. Jurik, A. Weaver, T. Blalock, and B. Calhoun, "A 2.6-uw sub-threshold mixed-signal ecg soc," in *VLSI Circuits, 2009 Symposium on*, June 2009, pp. 60–61.

[61] B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin, "A 2.60pj/inst subthreshold sensor processor for optimal energy efficiency," in *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*, 2006, pp. 154–155.

[62] C. Piguet, J.-M. Masgonty, S. Cserveny, C. Arm, and P. D. Pfister, "Low-power low-voltage library cells and memories," in *Electronics, Circuits and Systems, 2001. ICECS 2001. The 8th IEEE International Conference on*, vol. 3, 2001, pp. 1521–1524 vol.3.

[63] W. Cheng and B. Baas, "Dynamic voltage and frequency scaling circuits with two supply voltages," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 1236–1239.

[64] J. Kwong and A. Chandrakasan, "Variation-driven device sizing for minimum energy sub-threshold circuits," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, 2006, pp. 8–13.

[65] S. Inc., "Complete liberty documentation version 2008.09," Aug 2009. [Online]. Available: http://www.opensourceliberty.org/resources_ccs.html

[66] S. Hanson, M. Seok, D. Sylvester, and D. Blaauw, "Nanometer device scaling in subthreshold circuits," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 700–705.

[67] T. Sharma and R. Thilagavathy, "Performance analysis of advanced encryption standard for low power and area applications," in *Information Communication Technologies (ICT), 2013 IEEE Conference on*, 2013, pp. 967–972.

[68] B. Yu, X. Li, N. Zhang, and Y. Sun, "A low cost, low power aes asic with high dpa resisting ability," in *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, 2009, pp. 285–288.

[69] D. Wang, Y. Ding, J. Zhang, J. Hu, and H. Tan, "Area-efficient and ultra-low-power architecture of rsa processor for rfid," *Electronics Letters*, vol. 48, no. 19, pp. 1185–1187, 2012.

[70] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Ultra-lightweight cryptography for low-cost rfid tags: Hummingbird algorithm and protocol," 2009.

[71] M. Xiao, X. Shen, J. Wang, and J. Crop, "Design of a uhf rfid tag baseband with the hummingbird cryptographic engine," in *ASIC (ASICON), 2011 IEEE 9th International Conference on*, 2011, pp. 800–803.

[72] J. Tschanz, N. S. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vanga, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunaga, D. Somasekhar, S. Tang, D. Finan, T. Karnik, N. Borkar, N. Kurd, and V. De, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb 2007, pp. 292–604.

[73] K. Bowman et. al., "A 45 nm resilient microprocessor core for dynamic variation tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 194 –208, January 2011.

[74] S. Hsu, A. Agarwal, M. Anders, S. Mathew, H. Kaul, F. Sheikh, and R. Krishnamurthy, "A 280mv-to-1.1v 256b reconfigurable simd vector permutation engine with 2-dimensional shuffle in 22nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, 2012, pp. 178–180.

[75] J. Crop, S. Fairbanks, R. Pawlowski, and P. Chiang, "150mV sub-threshold asynchronous multiplier for low-power sensor applications," *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pp. 254–257, 2010.

[76] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. Gb, R. Ramanarayanan, V. Erraguntla, J. Howard, S. Vangal, S. Dighe, G. Ruhl, P. Aseron, H. Wilson, N. Borkar, V. De, and S. Borkar, "A 280mv-to-1.2v wide-operating-range ia-32 processor in 32nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, 2012, pp. 66–68.

[77] J. Crop, R. Pawlowski, N. Moezzi-Madani, J. Jackson, and P. Chaing, "Design automation methodology for improving the variability of synthesized digital circuits operating in the sub/near-threshold regime," in *Green Computing Conference and Workshops (IGCC), 2011 International*, 2011, pp. 1–6.

[78] R. Rithe, S. Chou, J. Gu, A. Wang, S. Datla, G. Gammie, D. Buss, and A. Chandrakasan, "Cell library characterization at low voltage using nonlinear operating point analysis of local variations," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, 2011, pp. 112–117.

[79] 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.

[80] E. Brunvand, S. Nowick, and K. Yun, "Practical advances in asynchronous design," in *Computer Design: VLSI in Computers and Processors, 1997. ICCD '97*, 1997, pp. 662–668.

[81] T. Werner and V. Akella, "Asynchronous processor survey," *Computer*, vol. 30, no. 11, pp. 67–77, 1997.

[82] O. Garnica, J. Lanchares, and R. Hermida, "Fine-grain asynchronous circuits for low-power high performance DSP implementations," *2000 IEEE Workshop on SiGNAL PROCESSING SYSTEMS. SiPS 2000. Design and Implementation (Cat. No.00TH8528)*, pp. 519–528, 2000.

[83] Y. Li, Z.-y. Wang, and K. Dai, "A low-power application specific instruction set processor using asynchronous function units," *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, pp. 817–822, Oct. 2007.

[84] M.-c. Chang and D.-s. Shiau, "Design of an asynchronous pipelined processor," *2008 International Conference on Communications, Circuits and Systems*, pp. 1093–1096, May 2008.

[85] O. Akgun, J. Rodrigues, and J. Sparsø, "Minimum-energy sub-threshold self-timed circuits: design methodology and a case study," in *2010 IEEE Symposium on Asynchronous Circuits and Systems*. IEEE, 2010, pp. 41–51.

[86] M. Singh, J. a. Tierno, A. Rylyakov, S. Rylov, and S. M. Nowick, "An adaptively pipelined mixed synchronous-asynchronous digital FIR filter chip operating at 1.3 Gigahertz," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 7, pp. 1043–1056, Jul. 2010.

[87] C. J. Myers, *Asynchronous circuit design*. Wiley, 2001.

[88] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[89] C. Pham-Quoc and A.-V. Dinh-Duc, "Hazard-free muller gates for implementing asynchronous circuits on Xilinx FPGA," *2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications*, pp. 289–292, 2010.

[90] S. Hauck, "Asynchronous design methodologies: an overview," *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69–93, 1995.

[91] G. Birtwistle and A. Davis, *Asynchronous digital circuit design*. Springer, 1995.

[92] J. Sparsøand S. Furber, Eds., *Principles of asynchronous circuit design: A systems perspective*. Kluwer Academic, Boston, 2001.

[93] S. Unger, "Asynchronous sequential switching circuits with unrestricted input changes," *IEEE Transactions on Computers*, vol. C-20, no. 12, pp. 1437–1444, Dec. 1971.

[94] M. Servit, "Hazard correction in asynchronous sequential circuits using inertial delay elements," *IEEE Transactions on Systems, Man, and Cybernetics*, May 1973.

[95] K.-L. Chang, B.-H. Gwee, and Y. Zheng, "A performance comparison on asynchronous matched-delay templates," *2009 IEEE International Symposium on Circuits and Systems*, pp. 1008–1011, May 2009.

[96] M. Renaudin, "The design of fast asynchronous adder structures and their implementation using D.C.V.S. Logic," *Circuits and Systems, 1994. ISCAS*, pp. 7–10, 1994.

[97] A. Martin, "Asynchronous datapaths and the design of an asynchronous adder," *Formal Methods in System Design*, pp. 119–137, 1991.

[98] D. Sokolov, "Automated synthesis of asynchronous circuits using direct mapping for control and data path," Ph.D. dissertation, University of Newcastle upon Tyne, 2005.

[99] K. Fant and S. Brandt, "NULL Convention Logic: a complete and consistent logic for asynchronous digital circuit synthesis," in *Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP '96*. IEEE Computer Soc. Press, 1996, pp. 261–273.

[100] N. Lotze, M. Ortmanns, and Y. Manoli, "A study on self-timed asynchronous subthreshold logic," in *Computer Design, 2007. ICCD 2007. 25th International Conference on*. IEEE, Oct. 2007, pp. 533–540.

[101] M. Ligthart, K. Fant, R. Smith, a. Taubin, and a. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," *Proceedings Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2000) (Cat. No. PR00586)*, pp. 114–125, 2000.

[102] S. C. Smith and W. K. Al-Assadi, "Teaching asynchronous digital design in the undergraduate computer engineering curriculum," *2007 IEEE Region 5 Technical Conference*, pp. 363–369, Apr. 2007.

[103] B. Reese, "Unified NULL Convention Logic Environment (uncle)." [Online]. Available: https://sites.google.com/site/asynctools/

[104] a. Branover, R. Kol, and R. Ginosar, "Asynchronous design by conversion: converting synchronous circuits into asynchronous ones," *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, pp. 870–875, 2004.

[105] J. Cortadella, a. Kondratyev, L. Lavagno, and C. Sotiriou, "Desynchronization: dynthesis of asynchronous circuits From synchronous specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1904–1921, Oct. 2006.

[106] M. Marshall and G. Russell, "A low power information redundant concurrent error detecting asynchronous processor," *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, no. Dsd, pp. 649–656, Aug. 2007.

[107] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *Asynchronous Architecture*, vol. 43, no. 5, pp. 301–307, Feb. 1996.

[108] S. Nowick, K. Yun, and P. Beerel, "Speculative completion for the design of high-performance asynchronous dynamic adders," *in Asynchronous*, pp. 210–223, 1997.

[109] M. E. Dean, D. L. Dill, and M. Horowitz, "Self-timed logic using Current-Sensing Completion Detection (CSCD)," in *Computer Design: VLSI in Computers and Processors, 1991. ICCD '91*, vol. 7, no. 1-2, Feb. 1991, pp. 187–191.

[110] O. Izosimov, I. Shagurin, and V. Tsylyov, "Physical approach to CMOS module self-timing," *Electronics Letters*, vol. 26, no. 22, pp. 835–836, 1990.

[111] E. Grass and S. Jones, "Asynchronous circuits based on multiple localised current-sensing completion detection," in *Asynchronous Design Methodologies, 1995. Proceedings., Second Working Conference on*, no. Cl. IEEE, 1995, pp. 170–177.

[112] M. J. Gamble, "A novel current-sensing completion-detection circuit adapted to the micropipeline methodology," M.Sc., University of Manitoba, 1995.

[113] H. Lampinen and O. Vainio, "Circuit design for current-sensing completion detection," in *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, vol. 2. IEEE, 1998, pp. 185–188.

[114] H. Lampinen, P. Perala, and O. Vainio, "Design of a self-timed asynchronous parallel FIR filter using CSCD," in *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, vol. 5. IEEE, 2003, pp. V–165.

[115] T. Kumaran, M. Santhi, M. Srikanth, N. Srinivasan, M. Balaji, and G. Lakshminarayanan, "Transient current sensing based completion detection with event separation logic for high speed asynchronous pipelines," in *TENCON 2009-2009 IEEE Region 10 Conference*. IEEE, 2009, pp. 1–6.

[116] L. Nagy and V. Stopjakova, "Current sensing completion detection in deep sub-micron technologies," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*. IEEE, 2010, pp. 145–148.

[117] B. Gadamsetti and A. Singh, "Current sensing completion detection for high speed and area efficient arithmetic," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*. IEEE, 2010, pp. 240–243.

[118] E. Grass and S. Jones, "Activity-monitoring completion-detection (AMCD); A new approach to achieve self- timing," *Electronics Letters*, vol. 32, no. 2, pp. 86–88, 1996.

[119] J. Crop, E. Krimer, N. Moezzi-Madani, R. Pawlowski, T. Ruggeri, P. Chiang, and M. Erez, "Error detection and recovery techniques for variation-aware CMOS computing: A comprehensive review," *Journal of Low Power Electronics and Applications*, vol. 1, no. 3, pp. 334–356, 2011. [Online]. Available: http://www.mdpi.com/2079-9268/1/3/334

[120] M. Keating, D. Flynn, R. Aitken, and K. Shi, *Low power methodology manual: for system-on-chip design*. Springer Verlag, 2007.

[121] K. Hu, T. Jiang, J. Wang, F. O'Mahony, and P. Chiang, "A 0.6 mw/gb/s, 6.4–7.2 gb/s serial link receiver using local injection-locked ring oscillators in 90 nm cmos," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 4, pp. 899 –908, Apr 2010.

[122] R. Pawlowski, E. Krimer, J. Crop, J. Postman, N. Moezzi-Madani, M. Erez, and P. Chiang, "A 530mv 10-lane simd processor with variation resiliency in 45nm soi," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, Feb 2012, pp. 492 –494.

[123] D. P. Siewiorek and R. S. Swarz, *Reliable computer systems (3rd ed.): design and evaluation*. Natick, MA, USA: A. K. Peters, Ltd., 1998.

[124] A. Avizienis, "Arithmetic error codes: cost and effectiveness studies for application in digital system design," *IEEE Transactions on Computers*, vol. C-20, pp. 1322–1331, 1971.

[125] N. Touba and E. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 16, no. 7, pp. 783–789, 1997.

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

[127] K. Mohanram, E. Sogomonyan, M. Gossel, and N. Touba, "Synthesis of low-cost parity-based partially self-checking circuits," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*. IEEE, 2003, pp. 35–40.

[128] H. Le, W. Starke, J. Fields, F. O'Connell, D. Nguyen, B. Ronchetti, W. Sauer, E. Schwarz, and M. Vaden, "IBM POWER6 microarchitecture," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 639–662, 2007.

[129] B. Sinharoy, R. Kalla, W. Starke, H. Le, R. Cargnoni, J. Van Norstrand, B. Ronchetti, J. Stuecheli, J. Leenstra, G. Guthrie *et al.*, "IBM POWER7 multicore server processor," *IBM Journal of Research and Development*, vol. 55, no. 3, pp. 1–1, 2011.

[130] J. H. Wensley, M. W. Green, K. N. Levitt, and R. E. Shostak, "The design, analysis, and verification of the SIFT fault tolerant system," in *ICSE '76: Proceedings of the 2nd international conference on Software engineering*, 1976, pp. 458–469.

[131] N. Oh, S. Mitra, and E. J. McCluskey, "Ed4i: Error detection by diverse data and duplicated instructions," *IEEE Trans. Comput.*, vol. 51, no. 2, pp. 180–199, 2002.

[132] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, and D. I. August, "SWIFT: software implemented fault tolerance," in *CGO '05: Proceedings of the International Symposium on Code Generation and Optimization (CGO'05)*, 2005, pp. 243–254.

[133] E. Mizan, T. Amimeur, and M. Jacome, "Self-imposed temporal redundancy: an efficient technique to enhance the reliability of pipelined functional units," in *Computer Architecture and High Performance Computing, SBAC-PAD 2007. 19th International Symposium on.* IEEE, 2007, pp. 45–53.

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

[135] ——, "Razorii: In situ error detection and correction for pvt and ser tolerance," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 32–48, 2009.

[136] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S.-L. Lu, T. Karnik, and V. De, "Energy-efficient and metastability-immune timing-error detection and instruction-replay-based recovery circuits for dynamic-variation tolerance," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 402 –623.

[137] S. Mitra, M. Zhang, N. Seifert, T. Mak, and K. Kim, "Soft error resilient system design through error correction," *VLSI-SoC: Research Trends in VLSI and Systems on Chip*, pp. 143–156, 2008.

[138] M. Dean, D. Dill, and M. Horowitz, "Self-timed logic using current-sensing completion detection (CSCD) ," *The Journal of VLSI Signal Processing*, vol. 7, no. 1, pp. 7–.16, 1994.

[139] F. Vargas and M. Nicolaidis, "SEU-tolerant SRAM design based on current monitoring," in *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on.* IEEE, 1994, pp. 106–115.

[140] E. Neto, I. Ribeiro, M. Vieira, G. Wirth, and F. Kastensmidt, "Using bulk built-in current sensors to detect soft errors," *Micro, IEEE*, vol. 26, no. 5, pp. 10–18, 2006.

[141] B. Gill, M. Nicolaidis, F. Wolff, C. Papachristou, and S. Garverick, "An efficient BICS design for SEUs detection and correction in semiconductor memories," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*. IEEE Computer Society, 2005, pp. 592–597.

[142] R. Troutman, *Latchup in CMOS technology: the problem and its cure*. Kluwer Academic Publishers, 1986.

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

[144] F. Tan and S. Fok, "Thermal management of mobile phone using phase change material," in *Electronics Packaging Technology Conference, 2007. EPTC 2007. 9th*, Dec 2007, pp. 836 –842.

[145] T. Maruyama, "SPARC64 VIII: Fujitsus next generation quad-core processor," in *Hot Chips*, vol. 20, 2008.

[146] S. Sankaran, J. Squyres, B. Barrett, V. Sahay, A. Lumsdaine, J. Duell, P. Hargrove, and E. Roman, "The lam/mpi checkpoint/restart framework: System-initiated checkpointing," *International Journal of High Performance Computing Applications*, vol. 19, no. 4, pp. 479–493, 2005.

[147] L. Rockett Jr, "An seu-hardened cmos data latch design," *Nuclear Science, IEEE Transactions on*, vol. 35, no. 6, pp. 1682–1687, 1988.

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

[149] T. Uemura, Y. Tosaka, H. Matsuyama, K. Shono, C. Uchibori, K. Takahisa, M. Fukuda, and K. Hatanaka, "Seila: Soft error immune latch for mitigating multi-node-seu and local-clock-set," in *Reliability Physics Symposium (IRPS), 2010 IEEE International*. IEEE, 2010, pp. 218–223.

[150] 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, 2000.

[151] G. Messenger, "Collection of charge on junction nodes from ion tracks," *Nuclear Science, IEEE Transactions on*, vol. 29, no. 6, pp. 2024–2031, 1982.

[152] D. Holcomb, W. Li, and S. Seshia, "Design as you see fit: System-level soft error analysis of sequential circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009, pp. 785–790.

[153] "Nangate 45nm open cell library," http://www.nangate.com.

[154] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Circuits and Systems, 1989., IEEE International Symposium on*. IEEE, 1989, pp. 1929–1934.

[155] A. Moody, G. Bronevetsky, K. Mohror, and B. de Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 2010, pp. 1–11.

[156] J. Chung, I. Lee, M. Sullivan, J. H. Ryoo, D. W. Kim, D. H. Yoon, L. Kaplan, and M. Erez, "Containment domains: a scalable, efficient, and flexible resilience scheme for exascale systems"," in *to appear in SC'12*, 2012.

[157] T. Hoffmann, G. Doorribos, I. Ferain, N. Collaert, P. Zimmerman, M. Goodwin, R. Rooyackers, A. Kottantharayil, Y. Yim, A. Dixit, K. De Meyer, M. Jurczak, and S. Biesemans, "Gidl (gate-induced drain leakage) and parasitic schottky barrier leakage elimination in aggressively scaled hfo/sub 2/tin finfet devices," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, Dec 2005, pp. 725 –728.