

ABSTRACT

Title of thesis: Reduction of Architecture Vulnerability
 Factor using Modified Razor Flip-flops

Kiran K Seshadri, Master of Science, 2007

Thesis directed by: Professor Manoj Franklin
 Department of Electrical and Computer Engineering

Microprocessors are increasingly being used in a variety of applications from small computers to life critical applications. With increasing dependence on microprocessor based systems in critical applications, greater importance needs to be given not only to performance but also to correctness and dependability. Research has shown that microprocessors and structures of the microprocessors such as microarchitectural logic state elements and memories are vulnerable to alpha particle impacts (Single Event Upsets) and timing errors which will lead to Soft errors and affect program correctness and reliability.

In this thesis, we have modified the Razor flip-flops and explored the use of these Modified Razor flip-flops in the Instruction Queue (IQ), the Reorder Buffer (ROB), the Level 1 Data Cache (DL1), the Load Queue (LQ) and the Store Queue (SQ) to increase their respective reliability and hence the overall reliability of the microprocessor. Modified Razor flip-flops detect soft errors and along with architectural modifications, ensure correctness of the microprocessor operation, thus resulting in decrease of vulnerability and increase of reliability. We have adopted

Architecturally Correct Execution (ACE) time based techniques to measure the Architecture Vulnerability Factor (AVF) of high performance microprocessors and their internal structures using the SPEC 2000 integer benchmarks. The contribution of individual bit-fields of the structures towards the overall AVF is computed, and the fields with higher contribution towards the overall ACE Time are identified. We have computed the reduction in AVF with the introduction of Modified Razor flip-flops for various combinations of fields that have high vulnerability. However, introduction of Modified Razor flip-flops results in higher area requirement on the die and higher power consumption. We have developed RTL models for IQ, ROB, LQ, and SQ to measure the increase in area and average power consumption. Area requirement and power consumption estimates for the data cache are done using Cacti-based techniques. We have identified the most cost-effective solution by identifying the fields of these microarchitectural structures - where Modified Razor flip-flops are introduced - that result in the highest percentage decrease in AVF per unit area-power product.

We observe that, by introducing Modified Razor flip flops selectively in fields with the highest percentage decrease in AVF per unit area-power (identified by our work as opcode and destination operand field of IQ, destination operand and destination operand value field of ROB, tag array of DL1, address field of LQ and SQ), the overall AVF of the micro-processor decreases by 32.23%, but the area requirement increases by 16.99% and the average power consumption increases by 10.33%.

REDUCTION OF ARCHITECTURE VULNERABILITY FACTOR
USING MODIFIED RAZOR FLIPFLOPS

by

Kiran Kalkunte Seshadri

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2007

Advisory Committee:
Professor Manoj Franklin, Chair/Advisor
Professor Bruce Jacob
Professor Donald Yeung

© Copyright by
Kiran Kalkunte Seshadri
2007

Dedication

I dedicate this thesis to my mother. Thank you for all your sacrifices for me.

Acknowledgments

I would like to thank my advisor, Professor Manoj Franklin for his guidance throughout my graduate studies. His advices, both professional and otherwise, were extremely valuable.

I owe my deepest thanks to my family - my parents, sisters, brothers, brother-in-law, aunts and friends who have always stood by me, guided and helped me through my life, my career and during the times when my life seemed difficult. Words cannot express the gratitude I owe them.

I thank Professor Rajeev Barua for giving me an opportunity to work in his research group. I would also like to thank Professor Bruce Jacob and Professor Donald Yeung for agreeing to serve on my thesis committee.

I would like to acknowledge help and support from the staff of the ECE Graduate office who were always ready to answer my queries.

I thank Professor Deepak Somaya of the R H Smith School of Business for having provided me with financial support and giving me an opportunity to assist him in his exciting research projects.

It is almost impossible to acknowledge everyone who have helped me reach this stage in life, and I apologize to those whom I've inadvertently left out.

Lastly, I thank God for having blessed me with so many good things in life.

Table of Contents

List of Tables	vi
List of Figures	vi
1 Introduction	1
1.1 Soft Errors and Single Event Upsets	1
1.2 Reliability, Vulnerability and Quantifying Vulnerability	2
1.3 Reducing Architecture Vulnerability Factor	2
1.4 Our Contribution	3
1.5 Organization of the Thesis	5
2 Soft Errors and Architecture Vulnerability Factor	6
2.1 Overview	6
2.2 Soft Errors and Impact on Industry	6
2.3 Silent Data Corruption and Detected Unrecoverable Errors	7
2.4 Measuring Soft Error Rates	8
2.5 Architecture Vulnerability Factor	9
3 Computation of Architecture Vulnerability Factor	11
3.1 Overview	11
3.2 ACE Bits and AVF	11
3.3 Identifying Control Path ACE and Non-ACE bits	12
3.4 Identifying Data Path ACE and Non-ACE Bits	14
3.5 Computation of ACE Time and AVF	15
4 Modified Razor Flip-Flops	18
4.1 Razor Flip-flops: Background	18
4.2 Modified Razor flip-flops	20
4.3 Reduction in AVF	23
4.4 Power and Area Overhead	24
5 Experimental Results	26
5.1 Simulation Model	26
5.2 RTL Models	27
5.3 AVF and ACE Time	29
5.4 Reduction in AVF with Modified Razor flip-flops	34
5.5 Increase in Area and Power	48
5.6 Cost effective Percentage decrease in AVF	49
5.7 Observations	51
6 Summary and Conclusion	56
6.1 Summary	56
6.2 Conclusion	57

List of Tables

3.1	Non-ACE fields of IQ and ROB for Instructions.	16
5.1	Processor Configuration.	27
5.2	Instructions skipped in SPEC 2000 Benchmarks.	28
5.3	Fields for Modified Razor flip-flops.	55

List of Figures

2.1	Soft Error Classification and Outcome.	8
3.1	Example of Cache Operation and ACE Time between States.	15
4.1	Razor Flip-flops between Logic Stages.	19
4.2	Razor Flip-flop Operation.	19
4.3	Modified Razor Flip-flop.	21
4.4	Modified Razor Flip-flop Operation.	22
4.5	Instruction Queue with Modified Razor Flip-flop in Valid, Operand and Destination Operand fields.	23
5.1	RTL Model of Instruction Queue with Modified Razor flip-flops.	29
5.2	RTL Model of Reorder Buffer with Modified Razor flip-flops.	30
5.3	RTL Model of Load/Store Queue with Modified Razor flip-flops.	30
5.4	AVF of Instruction Queue.	31
5.5	AVF of Reorder Buffer.	32
5.6	AVF of Level 1 Data Cache.	32
5.7	AVF of Load Queue.	33
5.8	AVF of Store Queue.	33

5.9	ACE Time distribution of Instruction Queue.	34
5.10	ACE Time distribution of Reorder Buffer.	35
5.11	ACE Time distribution of Level 1 Data Cache.	35
5.12	ACE Time distribution of Load Queue.	36
5.13	ACE Time distribution of Store Queue.	36
5.14	AVF of Instruction Queue with Modified Razor flip-flops.	37
5.15	AVF of Instruction Queue with Modified Razor flip-flops.	38
5.16	Percentage decrease in AVF of Instruction Queue with Modified Ra- zor flip-flops.	39
5.17	Percentage decrease in AVF of Instruction Queue with Modified Ra- zor flip-flops.	40
5.18	AVF of Reorder Buffer with Modified Razor flip-flops.	41
5.19	Percentage decrease in AVF of Reorder Buffer with Modified Razor flip-flops.	42
5.20	AVF of Level 1 Data Cache with Modified Razor flip-flops.	43
5.21	Percentage decrease in AVF of Level 1 Data Cache with Modified Razor flip-flops.	44
5.22	AVF of Load Queue with Modified Razor flip-flops.	45
5.23	AVF of Store Queue with Modified Razor flip-flops.	45
5.24	Percentage decrease in AVF of Load Queue with Modified Razor flip- flops.	46
5.25	Percentage decrease in AVF of Store Queue with Modified Razor flip- flops.	47
5.26	(a)Area and (b)Power consumption of Instruction Queue with Mod- ified Razor flip-flops.	49
5.27	(a)Area and (b)Power consumption of Reorder Buffer with Modified Razor flip-flops.	50
5.28	(a)Area and (b)Power consumption of Level 1 Data Cache with Mod- ified Razor flip-flops.	50

5.29	(a)Area and (b)Power consumption of Load/Store Queue with Modified Razor flip-flops.	51
5.30	Percentage decrease in AVF per Area-Power product of Instruction Queue with Modified Razor flip-flops.	52
5.31	Percentage decrease in AVF per Area-Power product of Reorder Buffer with Modified Razor flip-flops.	52
5.32	Percentage decrease in AVF per Area-Power product of Level 1 Data Cache with Modified Razor flip-flops.	53
5.33	Percentage decrease in AVF per Area-Power product of Load Queue with Modified Razor flip-flops.	53
5.34	Percentage decrease in AVF per Area-Power product of Store Queue with Modified Razor flip-flops.	54

Chapter 1

Introduction

Continuous improvement in CMOS technology has provided a steady increase in processor performance. However, due to scaling in feature size, voltage, density, microprocessors are vulnerable to *soft errors*. Soft errors are predicted to be the largest contributors of vulnerability of microprocessors and have emerged as a key challenge to reliability [1]. Technology trends have caused soft error rates in microprocessors to increase to levels that require changes to the design and implementation of present and future computing systems [2].

1.1 Soft Errors and Single Event Upsets

Soft errors are bit errors induced primarily due to Single Event Upsets (SEUs). The primary cause of SEUs are external radiations such as alpha particles and high energy neutrons from packaging materials, which trigger a change in the logic state of a semiconductor device. Secondary causes of soft errors include transient faults caused due to dynamic voltage scaling, voltage drops in the power supply networks, temperature fluctuations, and gate-length and doping concentration variations due to noise etc [3]. Soft errors do not cause or reflect a permanent fault in the device but introduce logical errors in the circuit's operation. Soft errors have made an impact on the industry and important microprocessor manufacturers because they

have caused random crashes in major customer sites [4].

1.2 Reliability, Vulnerability and Quantifying Vulnerability

Reliable operation and correctness are among the crucial concerns, apart from performance, for designing and manufacturing viable microprocessors. Soft errors are a major source of decrease in the reliability of microprocessors. Memory and microarchitectural logic state elements of the processors are vulnerable to soft errors. Techniques have been developed to estimate the soft error rates for various structures in microprocessors [1, 5]. Not all soft errors affect the correctness of the program outcome and hence it is important to quantify the vulnerability of structures to errors that result in user-visible program errors. Prior research has developed Architecturally Correct Execution (ACE) bit based analysis to measure Architecture Vulnerability Factor (AVF) as a measure of vulnerability [6]. AVF denotes the probability that a fault in a particular structure of the processor will result in a user-visible error [6]. Techniques also exist to compute AVF for address based structures [7].

1.3 Reducing Architecture Vulnerability Factor

A variety of techniques exist to prevent program incorrectness caused by soft errors and thus reduce the probability of generation of user-visible errors. Schemes such as Radiation hardened circuit design [8], localized error detection and correction [9], tri-modular [10] and bi-modular [11] redundant schemes, DIVA dynamic

verification [12], and checker process based schemes [13] have been developed to reduce the Architecture Vulnerability Factor. Each of the above methods reduce the vulnerability of the processor and its structures thereby reducing the Architecture Vulnerability Factor. Most of the schemes also introduce overheads in area, power, performance, and cost. Precise vulnerability estimates are required for designers to identify the correct tradeoff between reduction of vulnerability and cost.

1.4 Our Contribution

In this thesis, we have explored the use of Modified Razor flip-flops in the microarchitectural logic state elements of microprocessors viz. the Instruction Queue (IQ), the Reorder Buffer (ROB), the Data Cache (DL1), the Load Queue (LQ), and the Store Queue (SQ). Modified Razor flip-flops are designed to detect soft errors caused in microarchitectural logic state elements and storage structures within the microprocessor. Techniques involving Razor flip-flops were used to detect and correct timing errors in high-speed pipelines using delayed clocks [3]. We have modified the design of Razor flip-flops in order to be used in microarchitectural logic state elements and storage structures of any high performance out-of-order microprocessor. Soft errors, if any, will be detected by the Modified Razor flip-flops and hence the probability of generating user-visible program errors is minimized, which reflects in the reduction in the Architecture Vulnerability Factor of the processor. It is important to note that the introduction of Modified Razor flip-flops into the microprocessors would lead to increased area requirement and power consumption.

In our work, we have investigated the use of Modified Razor flip-flops in selected bit fields of the microarchitectural logic state elements and the data cache that will result in the highest percentage decrease in the overall AVF of the microprocessor with the least increase in the area requirement and power consumption. The rest of our effort is divided into three steps. First, we compute the AVF for microarchitectural logic state elements and the data cache using the out-of-order SimAlpha functional simulator and SPEC 2000 integer benchmarks. Further, we compute the contribution of each field of the IQ, ROB, DL1, LQ, and SQ towards the overall vulnerability of the respective structure so as to identify the fields of the microarchitectural logic state elements having high vulnerabilities. Second, we introduce our Modified Razor flip-flops for different combinations of the bit fields in each microarchitectural logic state element that would yield us the maximum reduction in overall AVF. As the third and final step of our work, for each of the combinations of fields identified in the second step, we compute the increase in area requirement and power consumption. We have developed RTL models to measure the exact increase in area requirement and power consumption for microarchitectural logic state elements. Area and power measurements for cache are done using Cacti [14]. We calculate the percentage decrease in AVF per unit area-power product for each of the microarchitectural logic state element and for the data cache to identify the best possible bit fields where Modified Razor flip-flops can be used and also remain cost effective.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 describes the background and terminology related to soft errors and Architecture Vulnerability Factor. Chapter 3 describes the AVF computation of the microprocessor and its microarchitectural logic state elements viz. the IQ, ROB, LQ, SQ and the DL1. Chapter 4 discusses Razor flip-flops and our modifications to the design of Razor flip-flops. Chapter 5 presents the experimental setup and the results and Chapter 6 summarizes and concludes our work.

Chapter 2

Soft Errors and Architecture Vulnerability Factor

2.1 Overview

This chapter describes the background and soft error terminology that is followed throughout the thesis. The chapter also discusses the background on Architecture Vulnerability Factor and computation of AVF for structures under consideration.

2.2 Soft Errors and Impact on Industry

Bit errors are mainly caused from energized particles such as neutrons from cosmic rays and alpha particles from packaging material, generating electron hole pairs as they pass through the semiconductor device. Transistor source and diffusion nodes can collect these charges. A sufficient amount of accumulated charge may invert the logic state of a device such as an SRAM cell, a gate, or a flip-flop [15]. Because these types of errors do not reflect a permanent failure of the device, they are termed as Soft errors. Soft error rates in microprocessors have become an increasing burden in the industry. One of the main reasons for the increase in soft errors and transient errors is a significant decrease in the feature size of transistors with new technologies facilitating the an exponential increase in the number of on-chip

transistors. Soft errors rates are also high in ultra low power microprocessors which use techniques like Dynamic Voltage scaling, that lead to reduction in operating and supply voltage levels.

Soft errors have caused a serious impact on the industry. There are a handful of documented events in the industry to substantiate the effect of soft errors. In the fifth generation SPARC64 processor, Fujitsu has protected 80% of the total 200K latches by parity check error detection, to counter cosmic ray strikes. The multiply/divide units are protected with residue check and parity prediction circuits [9]. Boeing Research has published several incidents of cosmic ray strikes in 1996 [16]. Sun Microsystems had acknowledged, in the year 2000, that cosmic ray strikes on unprotected L2 cache memories on UltraSPARC IIs caused its Ultra Enterprise servers to crash randomly at several customer sites (AOL, eBay, Verisign, and numerous other corporations were affected) [4]. Sun's UltraSPARC T1 processor has its integer and floating point register files protected by ECC, an extensive level of protection matched only by mainframe-class processors [17].

2.3 Silent Data Corruption and Detected Unrecoverable Errors

Figure 2.1 illustrates the possible outcomes of a single bit error. The flow chart also depicts the scenarios where errors are detected and corrected. From the flow chart it can be inferred that some bit errors do not cause a change in the program outcome. Detected Unrecoverable Errors (DUE) are those errors which cannot be corrected. DUEs can in turn be classified as True DUE and False DUE. As the

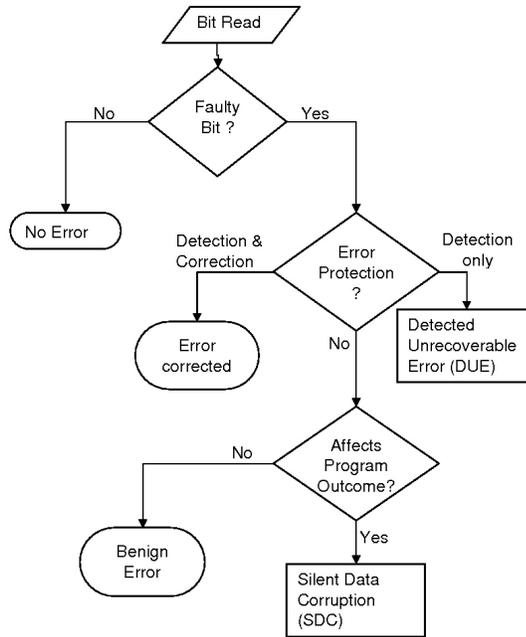


Figure 2.1: Soft Error Classification and Outcome.

names imply, True DUEs cause errors in the program outcome while False DUEs do not. Silent Data Corruption (SDC) is a scenario where a bit error causes a change in the program output, and is the most insidious form of errors. In order to enhance reliability, the errors have to be identified and corrected so as to prevent errors in program correctness.

2.4 Measuring Soft Error Rates

Currently the industry specifies soft error rates in terms of SDC and DUE numbers, and these are typically expressed using different metrics by vendors. Traditionally, Mean Time To Failure (MTTF) is used as the appropriate metric to measure system reliability. Other units commonly used are Failure In Time (FIT)

and Mean Time Between Failures (MTBF). MTBF is based on the interval between failures. MTBF of a component, as the name suggests, is the average time between failures. FIT is inversely related to MTBF. One FIT specifies one error in one billion operating hours. A zero error rate implies infinite MTBF and zero FIT. The overall FIT rate of a chip is calculated as the sum of the effective FIT rates of all the structures on-chip. Currently, typical FIT rate numbers for latches and SRAM cells vary from 0.001 FIT/bit to 0.01 FIT/bit at sea level and are projected to remain the same in the next several technology generations [18, 19, 16, 6].

2.5 Architecture Vulnerability Factor

Figure 2.1 illustrates that not all faults in a microarchitectural structure affect the final outcome of a program. To illustrate with an example, a bit error in a branch predictor will not affect the sequence or results of any committed instructions. Architectural Vulnerability Factor (AVF) is the probability that a fault in a processor structure will result in a visible error in the final output of a program. Thus, the branch predictor's AVF is 0%. In contrast, a single-bit fault in the committed program counter will cause the wrong instructions to be executed, almost certainly affecting the program's result. Hence, the AVF for the committed program counter is effectively 100%. Many structures will have an AVF that is in between these two extremes.

There are two important significance of AVF estimates. First, AVF gives the designers a measure of reliability of the processor and it does not depend on the

raw error rates which in turn depends on the two factors - application where the microprocessor is used and various environmental and fabrication factors. Second, the error rate of a microarchitectural logic state element is the product of its raw bit error rate and its AVF [6]. These overall error rates are used by designers and architects to ensure that over protection is not built into the structures for error detection and correction even where bit errors in these structures do not cause errors in program outcomes. For these structures, the estimated AVF will be low.

Chapter 3

Computation of Architecture Vulnerability Factor

3.1 Overview

This chapter describes the computation of AVF for the microarchitectural structures viz. the Instruction Queue (IQ), the Reorder Buffer (ROB), the Data Cache (DL1), the Load Queue (LQ), and the Store Queue (SQ). Also discussed is the accurate computation of the contribution of the bit fields of each of the above microstructural logic state elements towards the overall AVF of structures.

3.2 ACE Bits and AVF

The AVF of the overall structure is computed by analyzing each individual bit of every microarchitectural logic state element under consideration. Architecturally Correct Execution (ACE) bits will cause a visible error in the program outcome if a bit error is caused in them. Under real time analysis conditions, program outcome reflects the values conveyed to I/O structures. However, we define program outcome as the vales that get committed back to structures of the processor. Non-ACE bits are opposite to the ACE bits, which implies that a visible program error does not occur if a bit error is caused. The time a particular bit of a microarchitectural structure holds an ACE bit is defined as ACE Time or Residency of the Bit. The

AVF of the bit under consideration is the ratio of ACE Time to Total time for which the bit is analyzed. The AVF can equivalently be defined as the fraction of the time the microarchitectural storage bit under consideration can be classified as an ACE bit.

$$AVF \text{ of bit} = \frac{ACE \text{ Time of Bit (in Cycles)} \times 100}{Total \text{ Time (in Cycles)}} \quad (3.1)$$

The AVF of the microarchitectural structures like IQ, ROB, DL1, LQ and SQ is defined as

$$AVF \text{ of the structure} = \frac{[\sum_{\text{For all Bits}} ACE \text{ Time of Bit (in Cycles)}] \times 100}{Total \text{ Time (in Cycles)} \times Total \text{ Number of Bits}} \quad (3.2)$$

Equation 3.2 can be extended to all the microarchitectural structures of the microprocessor to compute the overall AVF of the microprocessor.

3.3 Identifying Control Path ACE and Non-ACE bits

An important aspect of computing AVF is to identify which of the microarchitectural storage bits are ACE and Non-ACE bits in the correct path instruction execution. All microarchitectural storage bits are conservatively assume as ACE bits unless proved otherwise. Architectural ACE and Non-ACE bits refer to the ACE and Non-ACE bits that are present in the correct path instruction execution. For example, an alpha particle strike on an IQ entry in the operand field of a NOP instruction will not affect the program outcome. Hence, corresponding bits of that IQ entry which holds the operand field can be considered as Non-ACE and the opcode field can be considered as ACE. The following four sources are considered for identifying ACE and Non-ACE bits.

- NOP instructions - Most instruction sets have NOP instructions which do not affect the architectural state of the processor. NOP instructions are inserted for a number of reasons such as to align instructions to address boundaries or to fill VLIW style instruction templates. Depending on the instruction set, the opcode field or the destination register specifier are the ACE bits for a NOP instruction and the remaining bits are considered Non-ACE.
- Performance enhancing instructions - Most modern processors include performance enhancing instructions like the prefetch instructions. A single bit error in a non-opcode field will cause wrong data to be fetched but the architectural state of the processor will not be affected. Hence, opcode or equivalent fields are ACE and the rest of the part of the is Non-ACE.
- Dynamically Dead instructions - Dynamically dead instructions are those whose results are not used by other instructions. Instructions whose results simply do not get read are called First-level Dynamically Dead (FDD) instructions. Transitively Dynamically Dead (TDD) are the instructions whose results are used only by the FDD instructions or other TDD instructions. The opcode and destination operand register specifier fields of the FDD and TDD are considered as ACE bits and the remaining fields are considered as Non-ACE.
- Trivial Instructions - Logical masking in the operand chains of Trivial instructions cause a source of Non-ACE bits under certain circumstances. Two types of Trivial instructions are identified. The first type of Trivial instructions have

two source register operands and the operand is masked with an immediate value. Depending on the type of masking and the value of the mask, some bits can be considered Non-ACE. For example, let us consider the following 2 instructions :

$$ORI\ R_A,\ R_B,\ 0xFF00$$

$$ANDI\ R_A,\ R_B,\ 0xFF00$$

The above ORI instruction performs a bitwise OR operation on the contents of the R_B register with 0xFF00. The 32-16 bits of the register R_B are Non-ACE because an alpha particle attack on these bits do not affect the architectural state. Similar arguments holds for the ANDI instruction. The second type of Trivial instructions have one source operand and immediate value. For example, we consider a multiply instruction with the source operand and an immediate value. If the value of the source operand is equal to 0, this will make the bits of the immediate value Non-ACE.

3.4 Identifying Data Path ACE and Non-ACE Bits

Computing AVF for address based structures such as the Data Cache, the Load Queue and the Store Queue is split into two parts [7].

- AVF and ACE Time for the data bits - Periods of time when the bit is Non-ACE state are identified. For example, Figure 3.1 shows the sequence of operations on a cache block. The cache data block can be considered Non-ACE in the period between two consecutive writes with no reads in between.

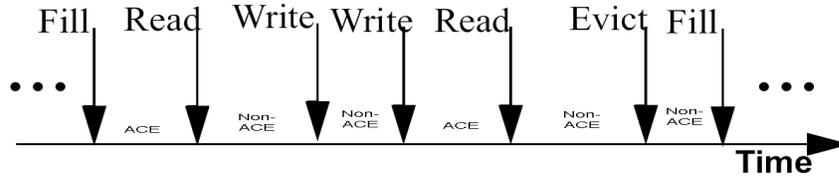


Figure 3.1: Example of Cache Operation and ACE Time between States.

Similar techniques are employed to identify Non-ACE bits for the data bits of the Load Queue and Store Queue.

- AVF and ACE Time for the tag bits - The second part involves the computation of ACE Time of the tag bits. The ACE Time of a tag array is measured by comparing the value of the tag array with the incoming tag bits. If the hamming distance between the incoming bits and the stored tag array value is 1 bit, the bits of that tag array are considered ACE because if a single bit error occurs in these tag entries, a cache hit will occur and hence would create a false positive and may cause an error in the program outcome. Conversely, if the hamming distance is more than 1, a false positive created by an alpha particle impact will not result in an error in the program outcome and hence the corresponding tag bits can be considered Non-ACE.

3.5 Computation of ACE Time and AVF

The ACE Time of a bit in a microarchitectural structure is the sum of all the cycles when the bit under consideration is ACE. ACE Bits and Non-ACE Bits are

identified for each cycle. ACE Time of a bit is the total number of cycles that the considered bit is ACE.

- ACE Time of IQ and ROB - Based on the type of instruction stored in the IQ, particular fields of the IQ are considered ACE or Non-ACE. The ACE Time of the IQ and the fields of the IQ are computed by multiplying the residency of an instruction in the IQ with the Number of ACE Bits for that instruction. For example, the ACE Time contribution by an instruction is given by the following equation -

$$ACE\ Time = (Total_{BW} - NonACE_{BW}) \times Residency\ (in\ Cycles) \quad (3.3)$$

where the

$Total_{BW}$ - Total Bit width

$NonACE_{BW}$ - Bit width of all NonACE Bits

Instruction Type	Non-ACE fields of IQ	Non-ACE fields of ROB
Dynamically Dead	All Source Operand register specifiers	Destination Operand Value
Prefetch and NOP	All fields except Status bits and Opcode	All fields except Status bits and Opcode
Trivial 1	One of the Source Operands	None
Trivial 2	Immediate Value or One of Source Operands	None

Table 3.1: Non-ACE fields of IQ and ROB for Instructions.

Table 3.5 enlists the different type of instructions and for each type of instruction, the fields of the IQ and ROB that are Non-ACE. All other fields are conservatively assumed to be ACE when the ACE Time contribution from the instruction is computed.

- ACE Time of Data cache, LQ, and SQ - The ACE time for the data array is calculated based on the lifetime analysis presented in the previous section. The number of cycles between the following state transitions for the data arrays of the LQ, SQ and the Level 1 Data Cache are considered Non-ACE - idle, fill-to-write, fill-to-evict, read-to-write, read-to-evict, write-to-write, write-to-evict, evict-to-fill. The ACE Time of the tag array is calculated by identifying all possible ACE tags for a particular incoming tag using the hamming distance analysis and adding the residency of the corresponding bits.

The AVF of microarchitectural structures can be computed using Equations 3.1 and 3.2 after ACE Time has been calculated. The overall AVF of the microprocessor can also be found by extending equation 3.2 for all the structures.

Chapter 4

Modified Razor Flip-Flops

In this chapter we discuss the design of Modified Razor flip-flops and the use of Modified Razor flip-flops to prevent Silent Data Corruption and reduce the AVF of the microprocessor.

4.1 Razor Flip-flops: Background

Razor flip-flops were proposed to detect and correct transient errors that are caused due to lowering voltage margins as a part of Dynamic Voltage Scaling (DVS) algorithms [3]. It is necessary to scale voltages as low as possible while ensuring correct operation of the processor. The critical voltage is chosen such that under a worst-case scenario of process and environmental variations, the processor always operates correctly. However, this approach leads to a very conservative supply voltage, because such a worst-case combination of different variabilities will be very rare. Razor flip-flops are introduced to detect and correct timing errors that are caused by lowering voltage margins to very low values. A Razor flip-flop double samples pipeline stage values, once with a fast clock and again with a delayed clock. A metastability-tolerant comparator then validates latch values sampled with the fast clock. In the event of a timing error, a modified pipeline mispeculation recovery mechanism restores correct program state.

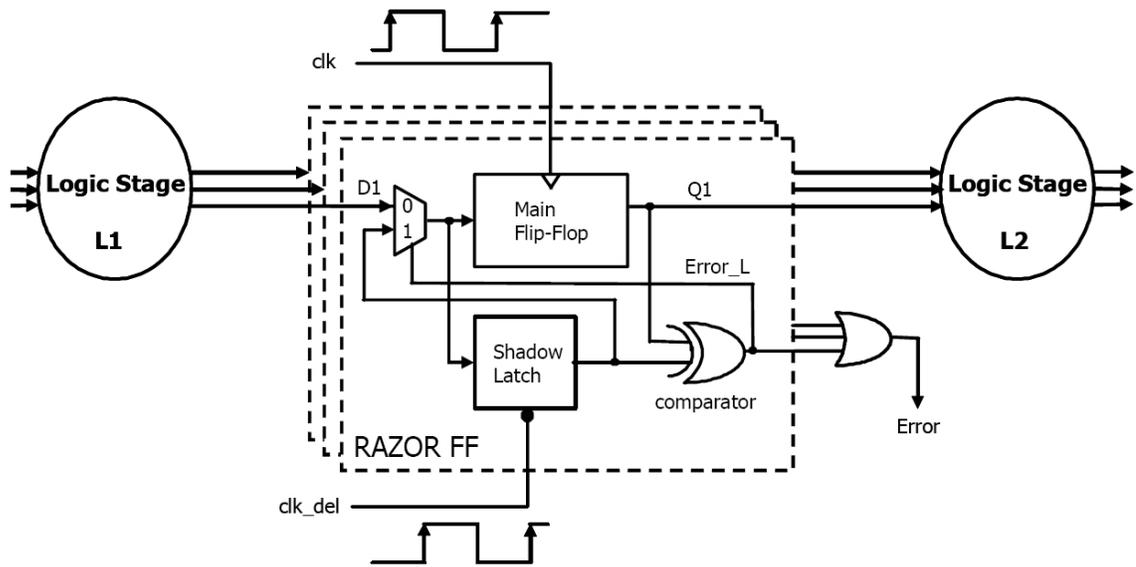


Figure 4.1: Razor Flip-flops between Logic Stages.

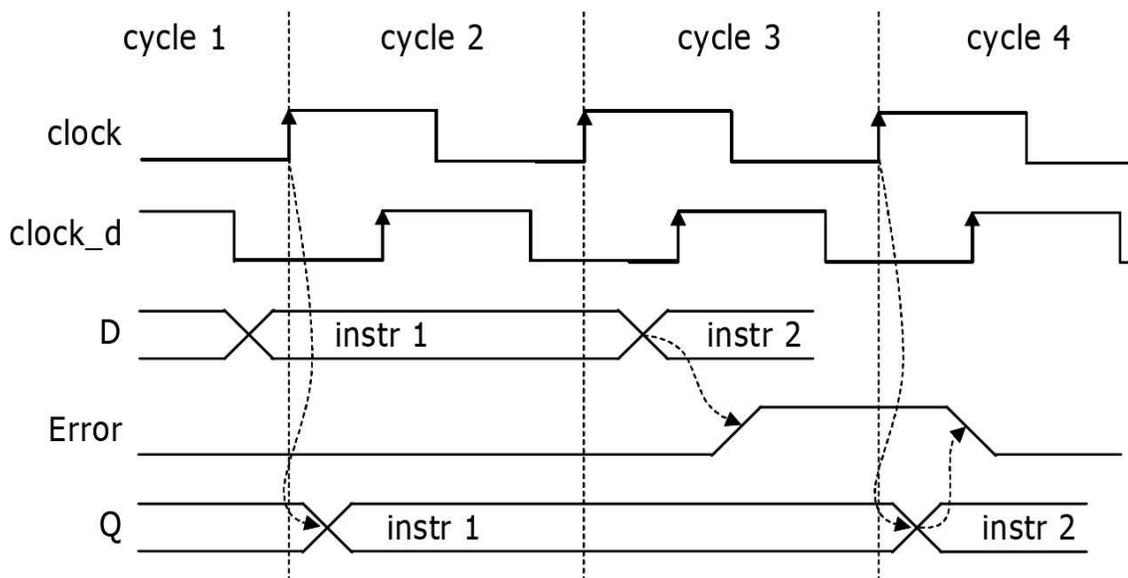


Figure 4.2: Razor Flip-flop Operation.

Razor relies on a combination of architectural and circuit level techniques for efficient error detection and correction of transient errors. The concept of Razor is illustrated in 4.1 for a pipeline stage. Each flip-flop in the design is augmented with a Shadow Latch which is controlled by a delayed clock. The operation of a Razor flip-flop is illustrated in 4.2.

In clock cycle 1, the combinational logic L1 meets the setup time by the rising edge of the clock and both the main flip-flop and the shadow latch will latch the correct data. In cycle 2, the combinational logic exceeds the intended delay due to subcritical voltage scaling and hence the data is not latched by the main flip-flop, but since the shadow-latch operates using a delayed clock, it successfully latches the data some time in cycle 3. By comparing the valid data of the shadow latch with the data in the main flip-flop, an error signal is then generated in cycle 3 and in the subsequent cycle, cycle 4, the valid data in the shadow latch is restored into the main flip-flop and becomes available to the next pipeline stage L2. The local error signals Error_L are ORed together to ensure that the data in all flip-flops is restored even when only one of the Razor flip-flops generates an error.

4.2 Modified Razor flip-flops

We have modified the design of the Razor flip-flops to be used for soft error detection in microarchitectural logic state elements viz. the IQ, the ROB, the LQ, and the SQ. An SRAM cell with similar error detecting capability is designed for the cache. Modified Razor flip-flops also rely on architectural and circuit level techniques

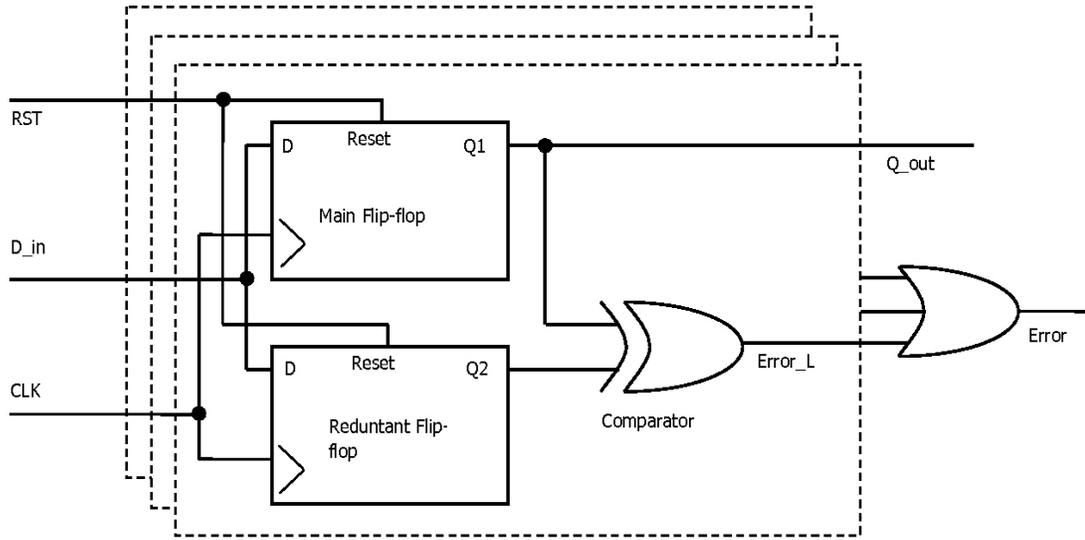


Figure 4.3: Modified Razor Flip-flop.

for efficient error detection and reduction in AVF of the microprocessor. The design of the Modified Razor flip-flop is shown in Figure 4.3. The logic state stored in a flip-flop is double sampled and stored in a redundant flip-flop as shown. At any given instance, both the flip-flops store the same logic state. If a soft error is caused due to an alpha particle impact, the logic state of one of the flip-flops changes. An XOR comparator is included in the Modified Razor flip-flop that will generate an Error signal. This operation is illustrated in Figure 4.4.

In clock cycle 1, the data from D_in is sampled in both the flip-flops - the Main flip-flop and Redundant flip-flop. In clock cycle 2, there is a change in the logic state of the Main flip-flop which may be caused by an alpha particle impact or any other cause of a soft error. When the logic state of the Main flip-flop changes, the local error signal Error_L is generated by the XOR comparator signaling an error in the

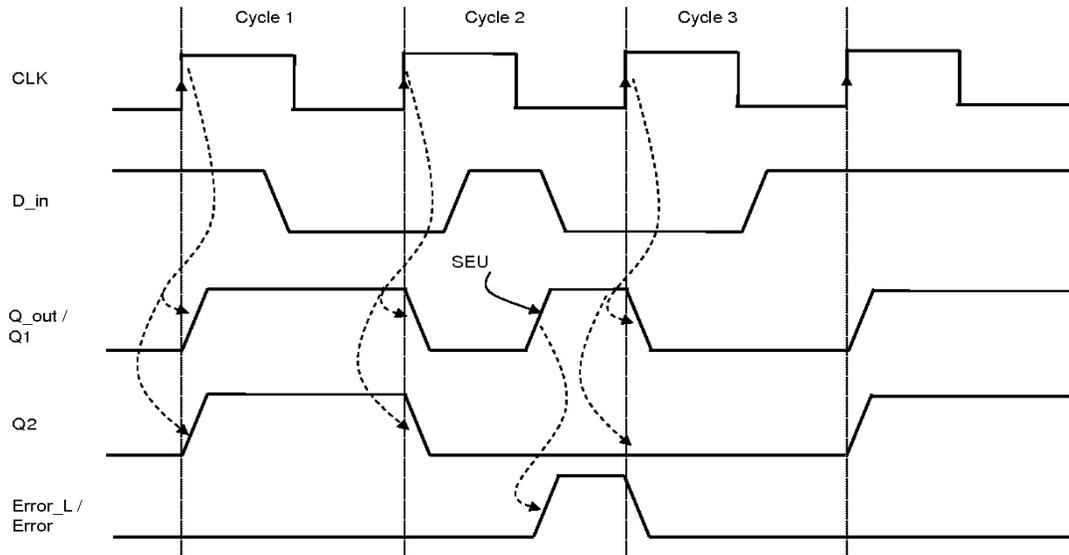


Figure 4.4: Modified Razor Flip-flop Operation.

logic state. Depending on the structure in which the Modified Razor flip-flop is used, architectural changes can be made such that the Error signal can be used to take appropriate actions to maintain program correctness. Figure 4.5 illustrates how the Modified Razor flip-flops can be used in the Opcode, Valid and Destination operand fields of one entry of the Instruction Queue. The Error signals from the Modified Razor flip-flops are used to reset the Valid bit of the IQ entry. Architectural changes to maintain correct program execution may include re-fetching the instruction for the IQ, re-fetching the data for the Data cache, re-executing certain instructions for the ROB, LQ, or the SQ.

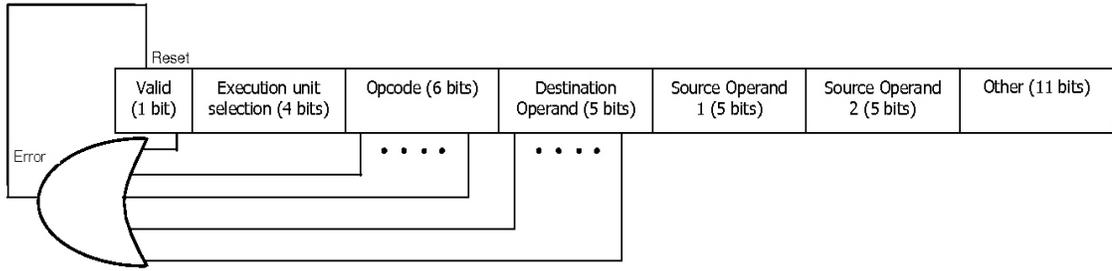


Figure 4.5: Instruction Queue with Modified Razor Flip-flop in Valid, Operand and Destination Operand fields.

4.3 Reduction in AVF

We have discussed AVF computation for microarchitectural structures in Chapter 3. The storage bits which are considered in the computation of AVF are considered always vulnerable to attack and a soft error in the bit may cause an error in the program outcome, hence increasing the Architecture Vulnerability Factor of the bit and also of the microprocessor. We have designed the Modified Razor flip-flops, which have the capability to detect errors caused in storage bits. The error detection capability of the Modified flip-flops and appropriate architectural techniques for error handling ensures that the program outcome is never erroneous. If a Modified Razor flip-flop is introduced into a storage bit instead of a normal flip-flop, we can safely conclude that, even with single event upsets, a soft error in that bit will never cause an error in the program outcome. Hence, that particular bit can always be considered as Non-ACE and the ACE Time contributed by that bit will be 0. The

equation for calculating the AVF of a microarchitectural element is

$$AVF = \frac{\left[\sum_{\text{For all non-Modified Razor FF Bits}} ACE \text{ Time of Bit (in Cycles)} \right] \times 100}{Total \text{ Time (in Cycles)} \times Total \text{ Number of Bits}} \quad (4.1)$$

Non-Modified Razor FF Bits in Equation 4.1 refers to those storage bits that do not have Modified Razor Flip-flops and hence do not have error detection capabilities.

The principle of AVF reduction is that when the Modified Razor flip-flops are used for storage bits that have high ACE Time contributions, the reduction in AVF is considerable, which we have shown in this thesis. In Chapter 5, we show how the AVF of the structures and also of the overall microprocessor reduce with the use of Modified Razor flip-flops in the IQ, ROB, DL1, LQ. and the SQ.

4.4 Power and Area Overhead

Modified Razor flip-flops lead to increased power consumption and area overhead due to built in error detection capabilities. Increase in area is due to the redundant flip-flop and the comparator that enable error detection capabilities. Increased area can also result in additional width of current driving elements which drive two flip-flops - main and redundant flip-flops. Increased power consumption is mainly due to the doubling of capacitance on the line that drives the input to the Modified Razor flip-flops. Other forms of power due to static and leakage currents also increase with introduction of an additional flip-flop. In Chapter 5, we have quantified the increase in Area and Power with the introduction of Modified Razor flip-flops into microarchitectural logic state elements viz. the IQ, the ROB, the the

DL1, the LQ and the SQ.

Chapter 5

Experimental Results

5.1 Simulation Model

For microarchitectural simulations, the SimSODA simulator, which is based on the SimAlpha 2.0, is used [20, 21]. We have modified the SimSODA simulator to introduce the capability to obtain ACE Time and AVF estimates for each individual field of the microarchitectural structures. The IQ, ROB, LQ and SQ were modified and the fields of these structures were modeled. The techniques for computation of the ACE Time for these fields were added into the model. The IQ, ROB, LQ, and SQ are modeled based on the microarchitecture of the Compaq DEC Alpha 21264 processor [22]. An out-of-order processor is simulated to obtain the AVF estimates. Table 5.1 enlists the significant system parameters of the simulated processor.

For simulations, the SPEC 2000 Integer Benchmark suite is used [23]. The following integer benchmarks are used : bzip2, crafty, eon, gap, gcc, gzip, mcf, parser, perlbnk, twolf, vortex and vpr. We have concentrated our efforts on the integer benchmarks as the simulation with SimAlpha for integer benchmarks are reported to be more accurate than the floating point benchmarks [21, 24]. To reduce simulation time while still maintaining representative program behavior, SimPoint analysis is used on the twelve integer points selected [25]. Table 5.1 presents the number of instructions to be skipped and the input data set selected. Each benchmark is then

System Parameter	Value
ROB Size	80 entries
Integer IQ Size	20 entries
Floating Point IQ Size	15 entries
LQ Size	32 entries
SQ Size	32 entries
L1 Cache : Split I and D	64KB, 2 way set associative
Fetch Width	4 instruction/cycle
Decode Width	4 instruction/cycle
Integer Issue Width	4 instruction/cycle
Floating Point Issue Width	2 instruction/cycle
Functional Units	4 Integer Execution Units 1 Floating Point Exec Units
Cluster	1 Integer Cluster 1 Floating Point Cluster

Table 5.1: Processor Configuration.

simulated for 100 million instructions.

5.2 RTL Models

We developed RTL models for the IQ, ROB, LQ and SQ to accurately measure the increase in area and power. We used Verilog 2000 to design the RTL models. Synopsys Design Compiler is used for simulation, synthesis and analysis of the design

Benchmark	Instructions skipped in Millions
bzip2-source	63
crafty	124
eon-rushmeier	215
gap	88
gcc-166	30
gzip-graphic	1
mcf	143
parser-dict	1771
perlbmk-splitmail	1
twolf	312
vortex-3	47
vpr-route	3

Table 5.2: Instructions skipped in SPEC 2000 Benchmarks.

of the structures [26]. The lsi_10k ASIC standard cell library is used for synthesis. Power and area reports from the Synopsys Design Compiler are obtained by choosing the highest level of optimization and highest detail of analysis. Figures 5.1, 5.2 and 5.3 depict the Register Transfer level design of the microarchitectural structures under consideration and the fields of each of the structures.

Figure 5.1 illustrates the structure of the Instruction Queue. The IQ is modeled to have 20 entries with each entry having a bit width to hold one instruction (which

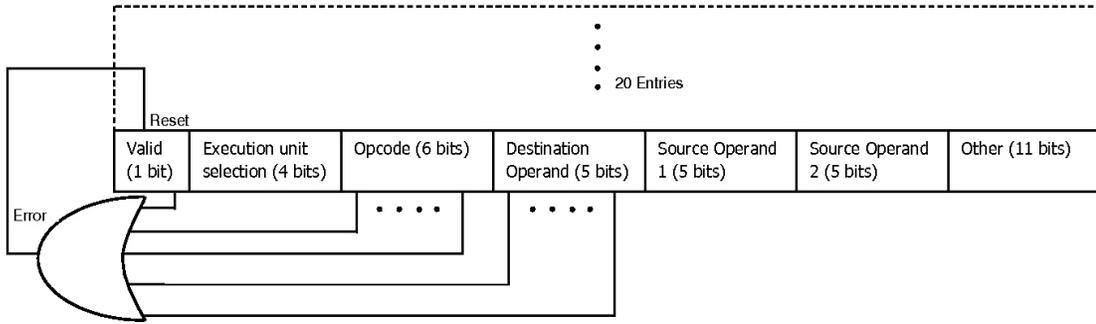


Figure 5.1: RTL Model of Instruction Queue with Modified Razor flip-flops.

includes the opcode, destination and source operand register specifiers) and other bits which involve execution unit selection bits and a Valid bit. The error signals from the Modified Razor flip-flops of an IQ entry are ORed and is used to reset the Valid bit of that IQ. The model for the ROB is similar to the IQ model and is illustrated in Figure 5.2. Each entry of the ROB has the following fields : the valid and complete bits, the program counter, the opcode bits, the destination operand register specifier and the destination operand value. The ROB is modeled to have 80 entries. The model for the LQ/SQ is shown in figure 5.3. Each entry of the LQ/SQ have valid and complete bits, destination/source address bits and destination/source value bits. The LQ/SQ is modeled to have 32 entries.

5.3 AVF and ACE Time

AVF computation is done using the techniques described in Chapter 3. The ACE Time for all the fields are computed. The residency of each committed instruction in a given structure is measured and accumulated to obtain the ACE Time of

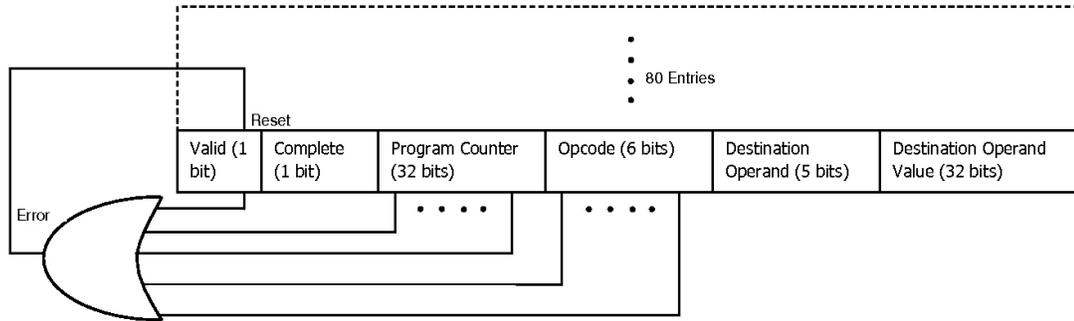


Figure 5.2: RTL Model of Reorder Buffer with Modified Razor flip-flops.

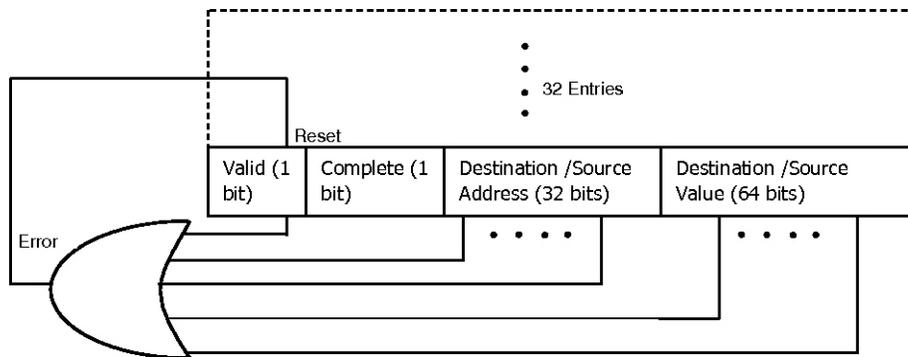


Figure 5.3: RTL Model of Load/Store Queue with Modified Razor flip-flops.

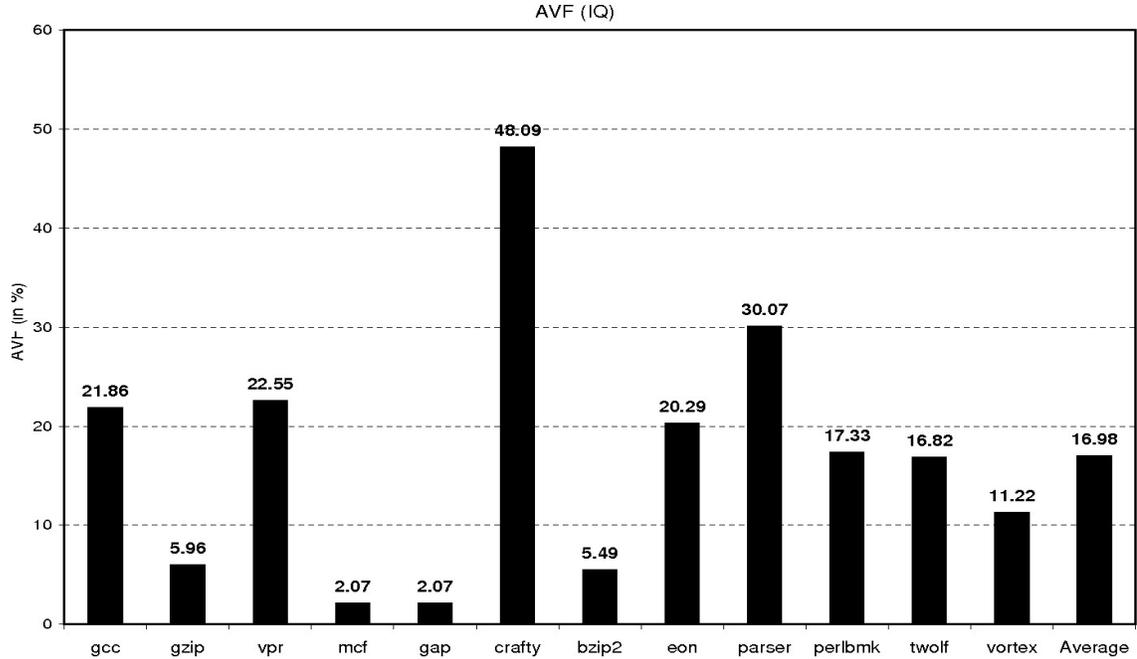


Figure 5.4: AVF of Instruction Queue.

the structure. Figures 5.4, 5.5, 5.6, 5.7 and 5.8 show the AVF values for the IQ, the ROB, the DL1, the LQ and the SQ respectively. Each graph shows the AVF for all SPEC 2000 integer benchmarks. We see that the Average AVFs of IQ, ROB, DL1, LQ and SQ are 16.98%, 30.37%, 31.66%, 21.65% and 29.28% respectively.

In order to understand the contribution of ACE Time from the individual fields of the structures towards the overall AVF, we look the ACE Time contribution from the individual fields for each structure. Figures 5.9, 5.10, 5.11, 5.12 and 5.13 show the ACE Time distribution from the IQ, the ROB, the DL1, the LQ, and the SQ. From the graphs below, we see that Average ACE Time contributions of Operand field of IQ (37.72%), Program counter field of ROB (61.68%), Data Array of Data Cache (67.61%), Address field of Load Queue (56.08%) and Address field of Store

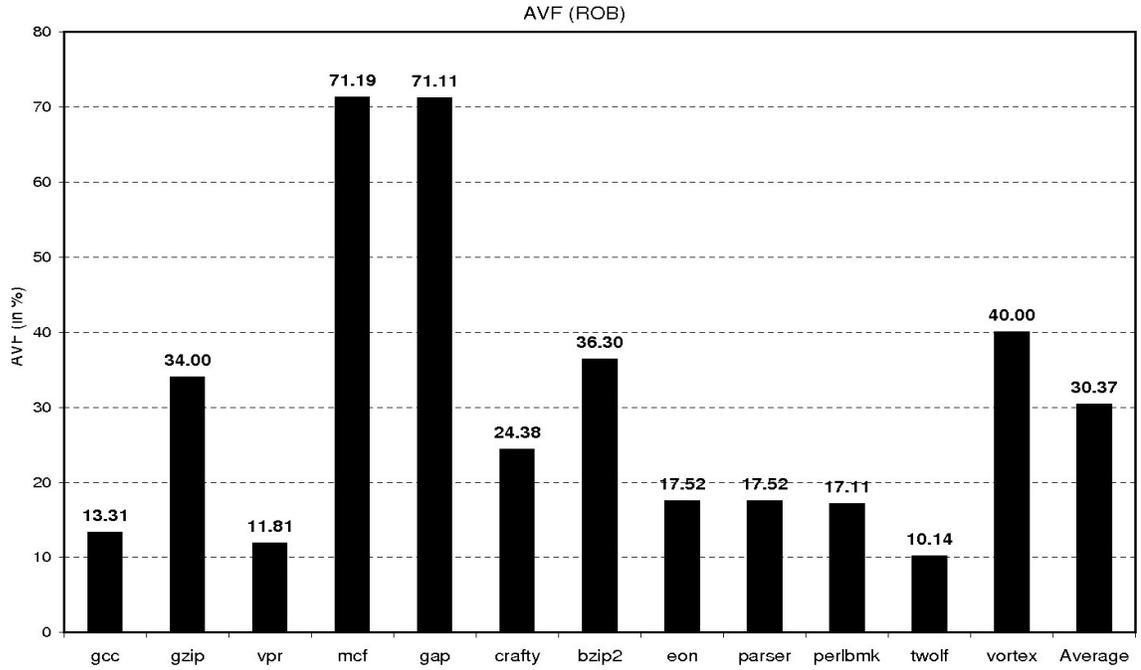


Figure 5.5: AVF of Reorder Buffer.

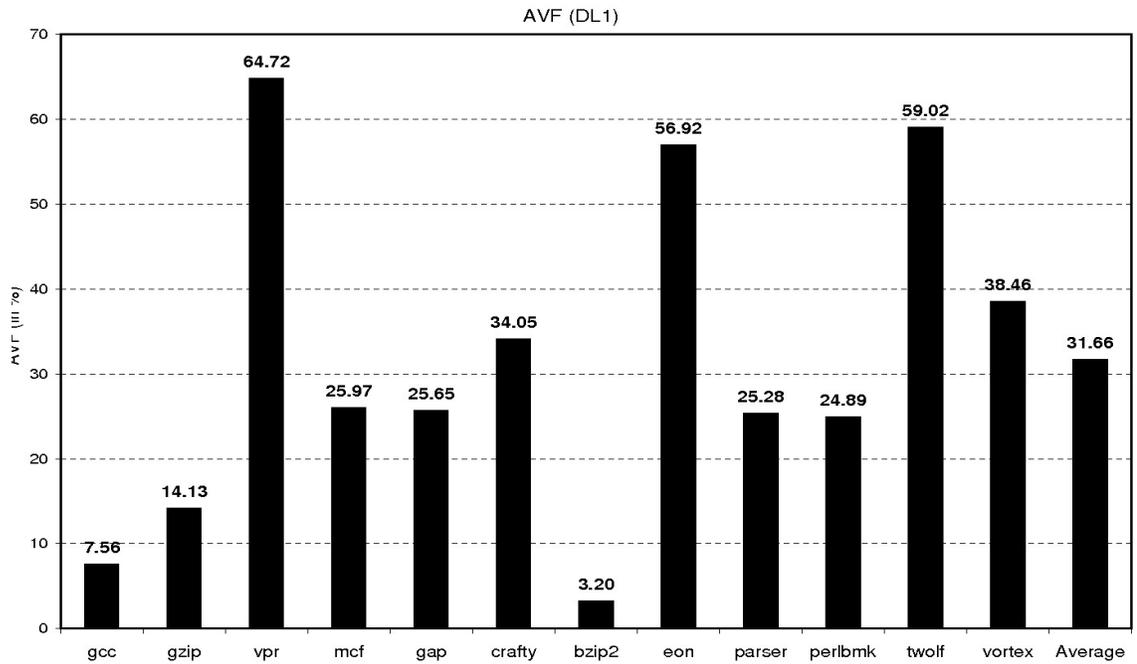


Figure 5.6: AVF of Level 1 Data Cache.

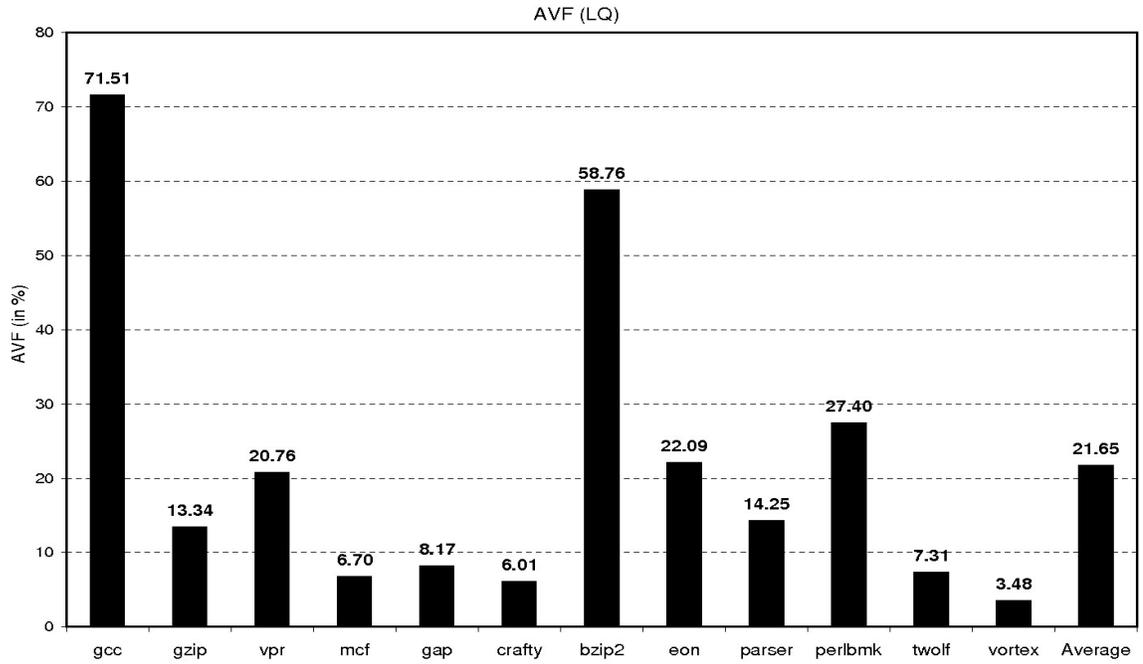


Figure 5.7: AVF of Load Queue.

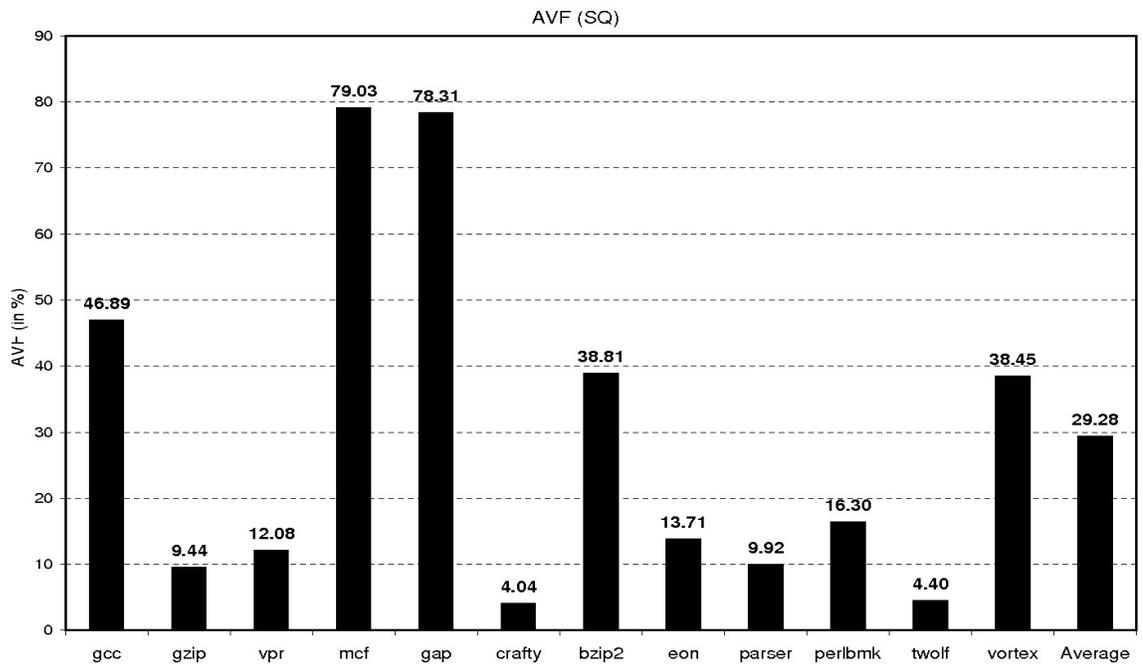


Figure 5.8: AVF of Store Queue.

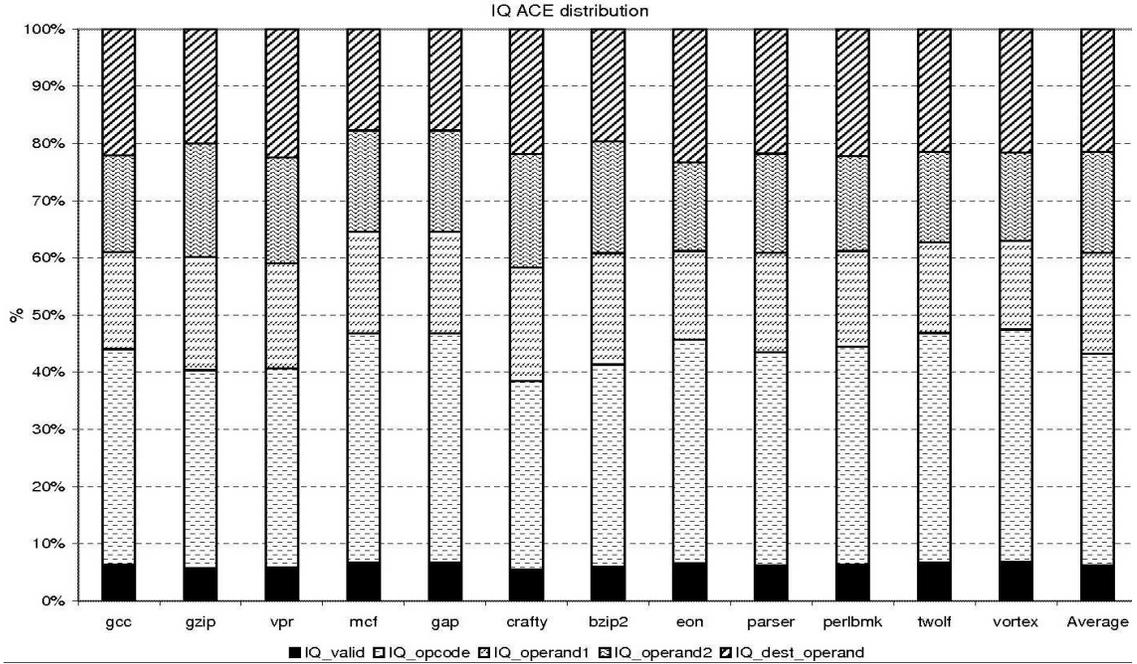


Figure 5.9: ACE Time distribution of Instruction Queue.

Queue (59.35%) towards their respective structures are significant. As these fields have significant contributions towards overall AVF and towards the AVF of their respective structures, we introduce Modified Razor flip-flops into these fields and into a combination of fields which high ACE Time contributions.

5.4 Reduction in AVF with Modified Razor flip-flops

Figures 5.9, 5.10, 5.11, 5.12 and 5.13 illustrate the contributions of each individual field towards the ACE Time of the overall structure, which in-turn is the reflection of the contribution of each field towards the AVF of the structure. We saw that the contributions of some fields are significant and we introduce the Modified Razor flip-flops into these fields to decrease the AVF. The AVF is calculated for

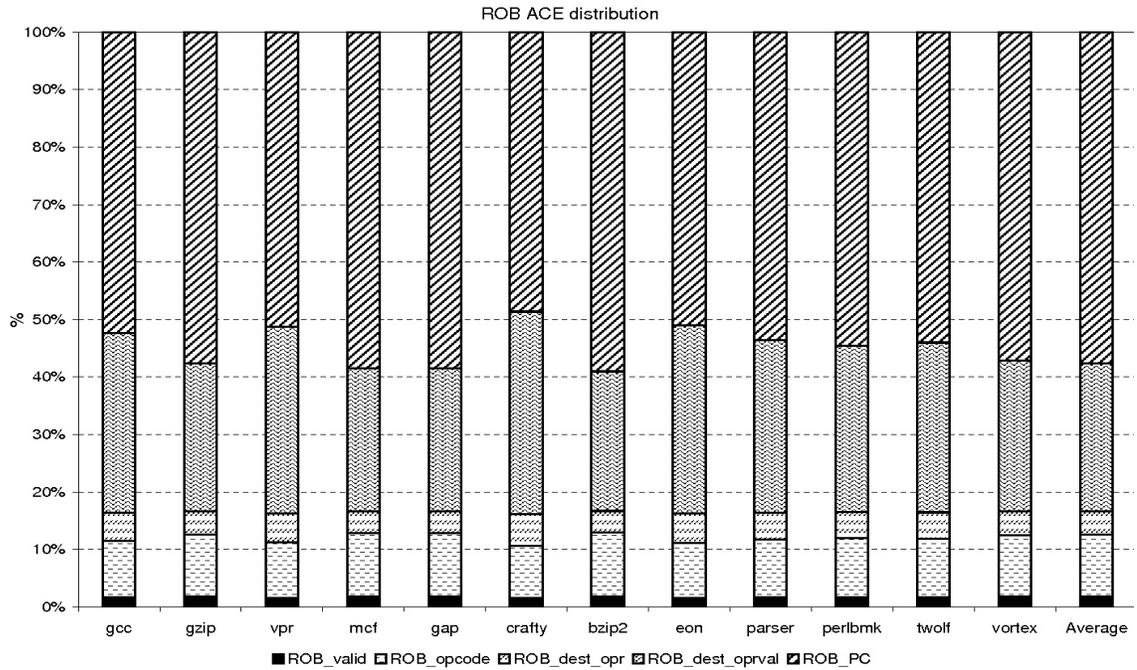


Figure 5.10: ACE Time distribution of Reorder Buffer.

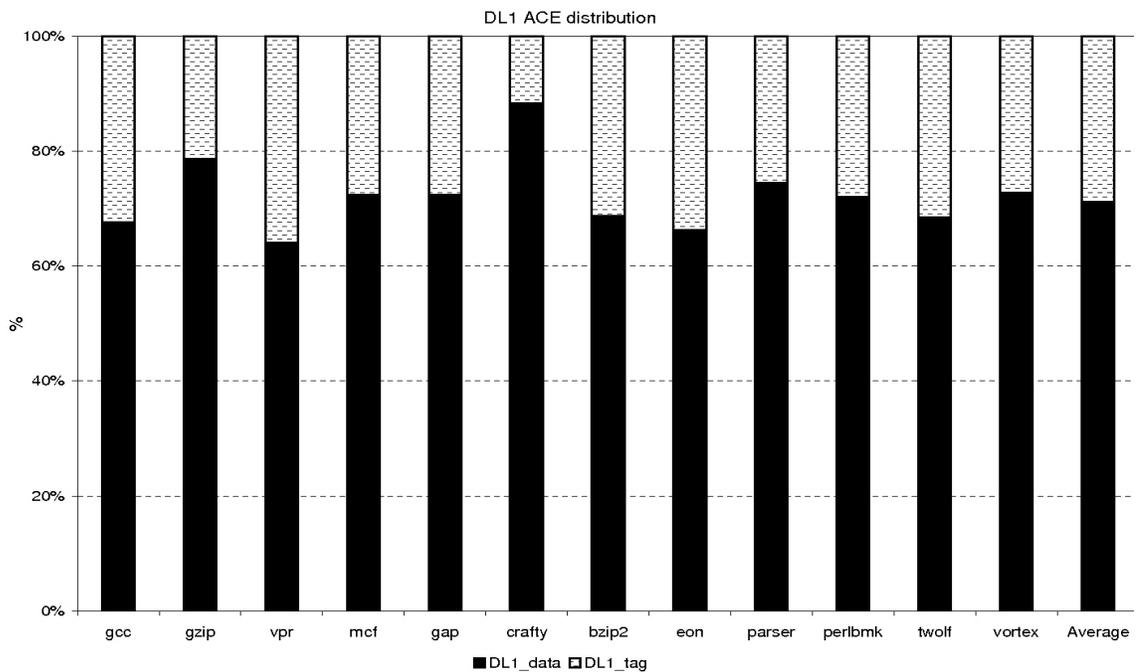


Figure 5.11: ACE Time distribution of Level 1 Data Cache.

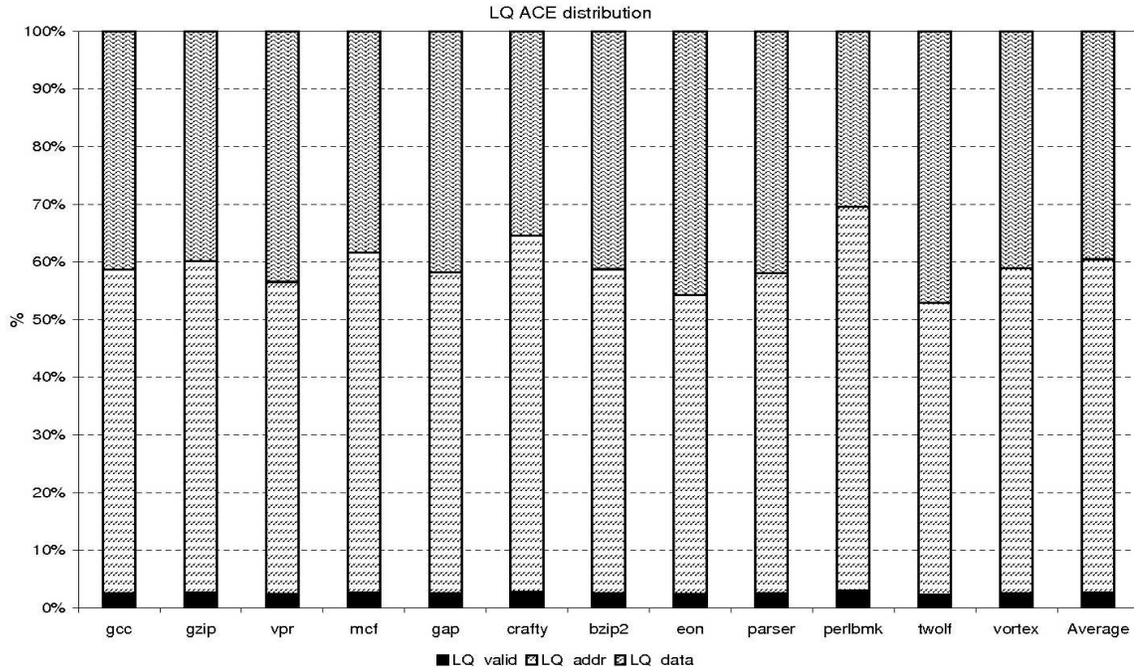


Figure 5.12: ACE Time distribution of Load Queue.

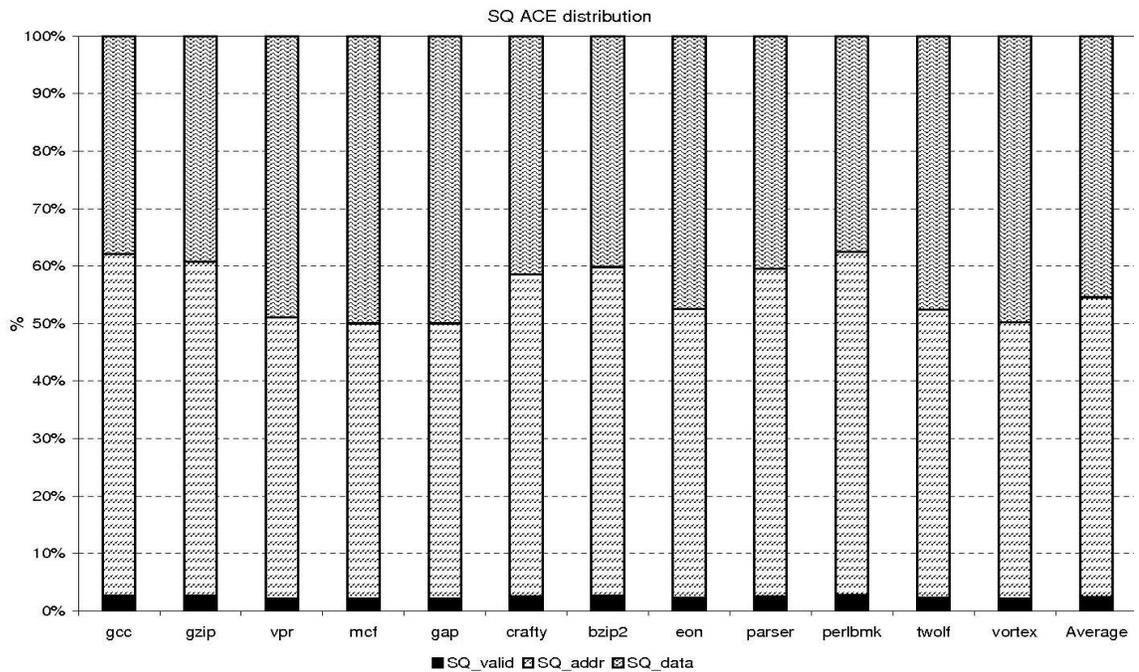


Figure 5.13: ACE Time distribution of Store Queue.

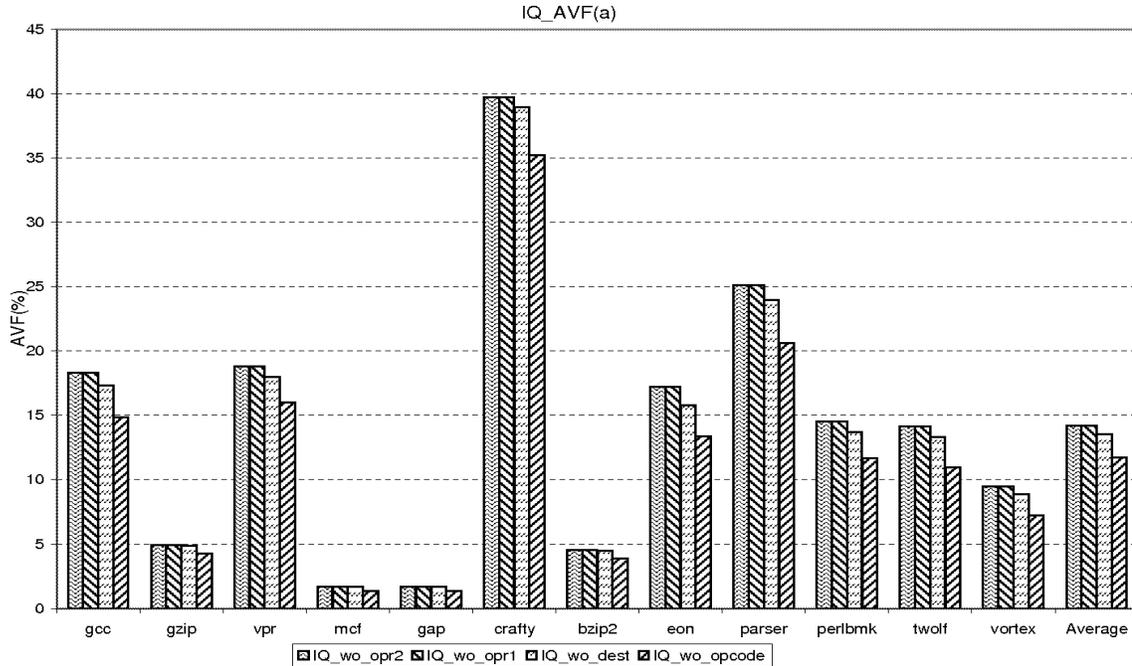


Figure 5.14: AVF of Instruction Queue with Modified Razor flip-flops.

the structures with Modified Razor flip-flops as presented in the Chapter 4. We have computed the reduction in the AVF of the structures when our Modified Razor flip-flops are introduced in fields that have a high contribution of ACE Time. In order to reduce AVF further, we have explored the introduction of these Modified Razor flip-flops in more than one fields.

Figures 5.9 shows a scenario where our Modified Razor flip-flops are introduced in the various fields of the IQ. We introduce Modified flip-flops in the opcode field, the destination and source operand fields and combinations of these fields. In Figure 5.14, we see that Modified Razor flip-flops are introduced in the Opcode field (IQ_wo_opcode), the Source and Destination Operand fields (IQ_wo_opr1, IQ_wo_opr2 and IQ_wo_dest). Figure 5.16 shows the percentage decrease in the

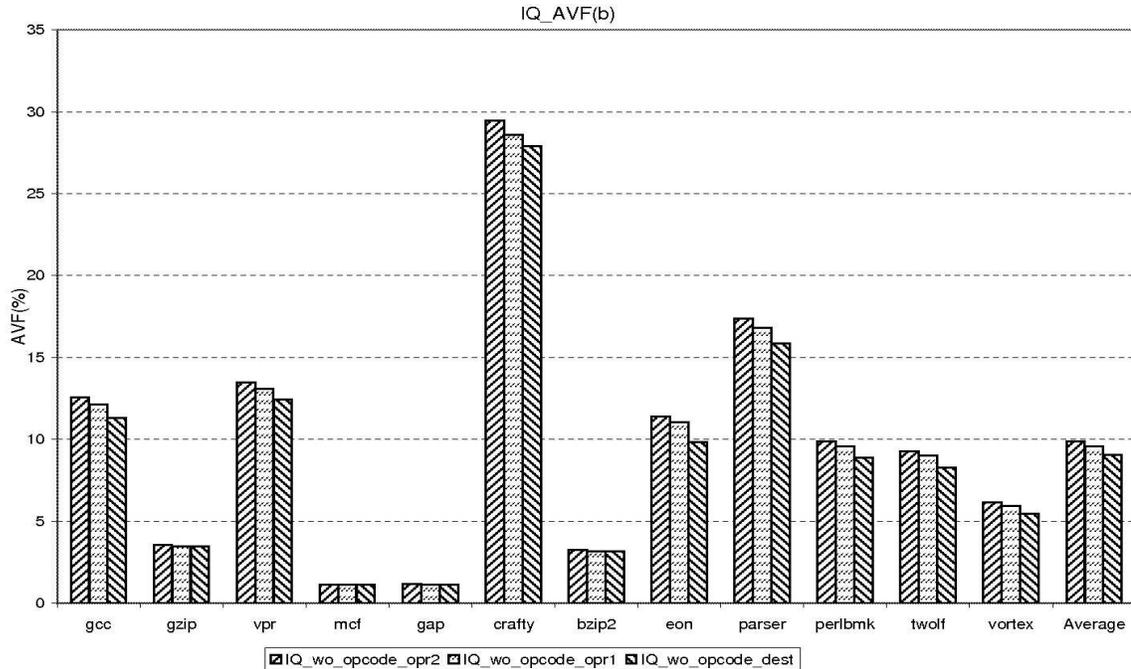


Figure 5.15: AVF of Instruction Queue with Modified Razor flip-flops.

AVF of the IQ for each case. To further decrease the AVF, we introduce our Modified Razor flip-flops in combinations of the above fields. Figure 5.14 shows the reduced AVFs when the Modified Razor flip-flops are introduced in the Opcode and Source Operand 2 fields (IQ_wo_opcode_opr2), the Opcode and Source Operand 1 fields (IQ_wo_opcode_opr1) and the Opcode and Destination Operand fields (IQ_wo_opcode_dest). Figure 5.17 shows the percentage reduction in AVF when the Modified Razor flip-flops are introduced in the combination of fields of the IQ. We see that when we introduce the Modified Razor flip-flops in Opcode and Destination Operand fields, the average AVF reduces to 9.06%, which is a 46.87% reduction from the initial value.

Similar analysis is done for the ROB as shown in Figures 5.18 and 5.19.

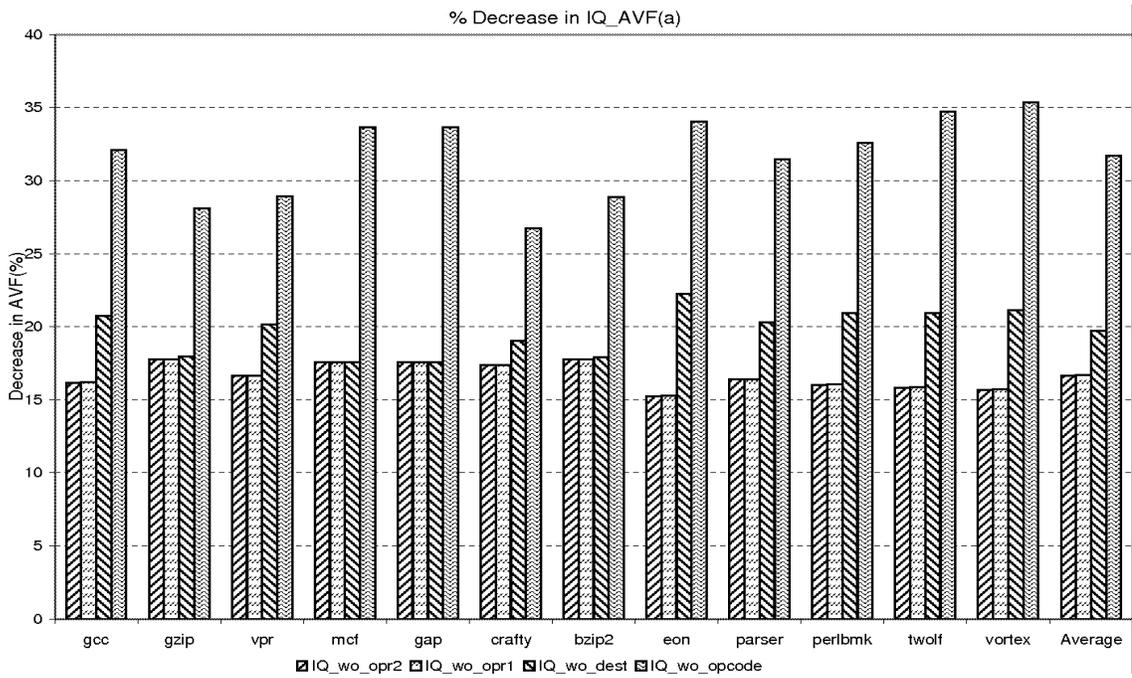


Figure 5.16: Percentage decrease in AVF of Instruction Queue with Modified Razor flip-flops.

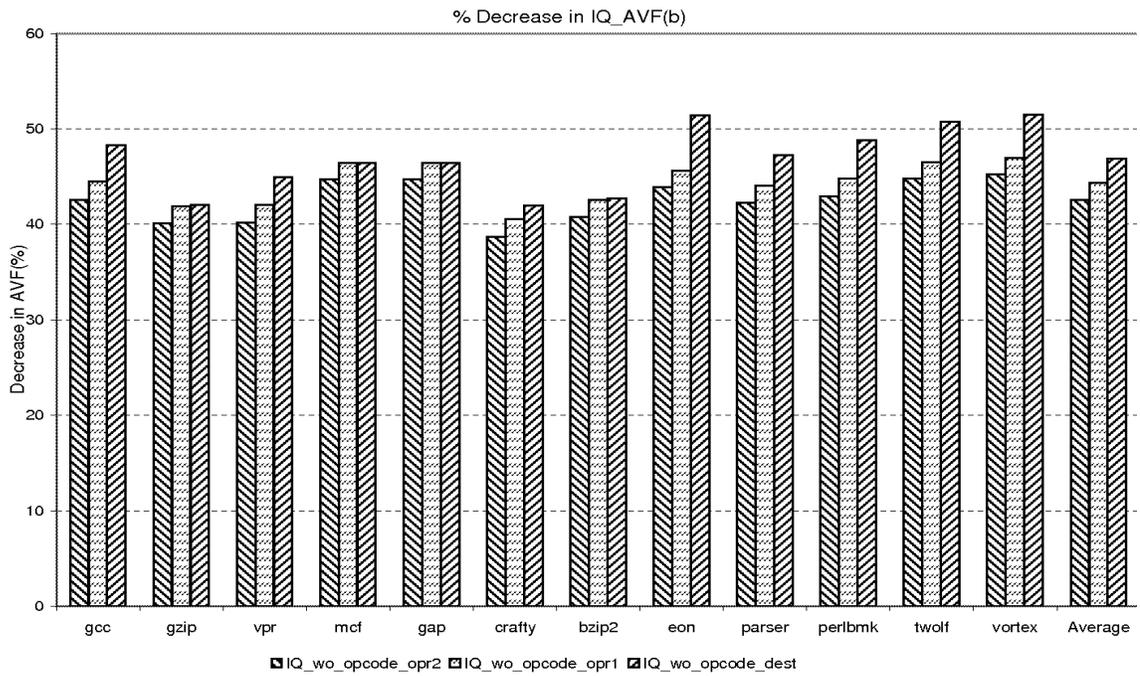


Figure 5.17: Percentage decrease in AVF of Instruction Queue with Modified Razor flip-flops.

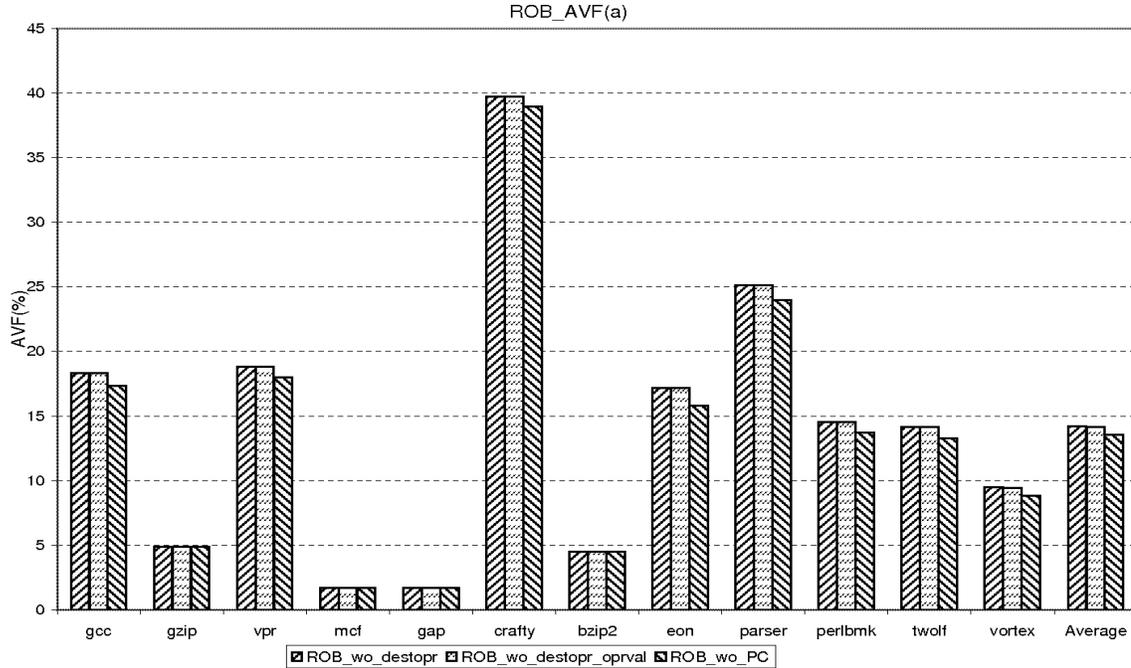


Figure 5.18: AVF of Reorder Buffer with Modified Razor flip-flops.

Modified Razor flip-flops are introduced in the following fields - the Destination Operand register specifier field (ROB_wo_destopr), the Destination Operand Value field (ROB_wo_destopr_value) and the Program Counter field (ROB_wo_PC). We see that when our Modified Razor flip-flops are introduced in the Program Counter field of the ROB, the average AVF reduces by 61.88% and the reduced value of the average AVF is 11.58%.

Figures 5.20 and 5.21 show the reduced average AVFs and percentage reduction in average AVFs when the Modified Razor flip-flops are introduced in the following fields of the Data Cache - Tag array (DL1_wo_tag) and the Data array (DL1_wo_data). We see that the maximum reduction in the AVF of the data cache is when our Modified Razor flip-flops are introduced in data array. The average

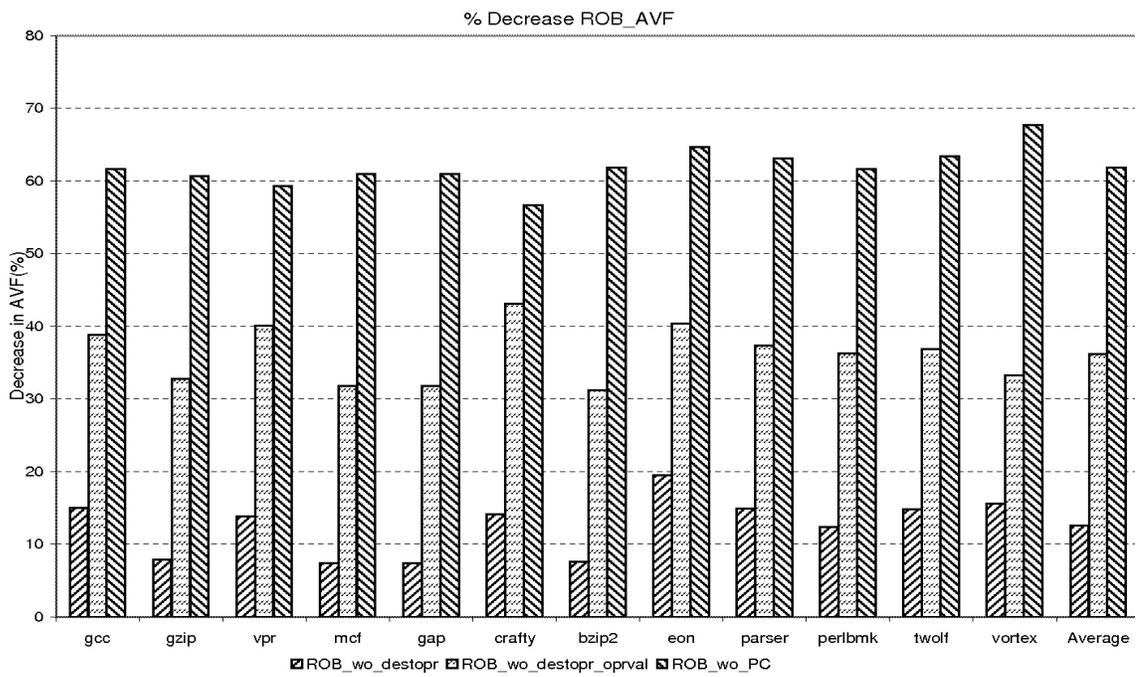


Figure 5.19: Percentage decrease in AVF of Reorder Buffer with Modified Razor flip-flops.

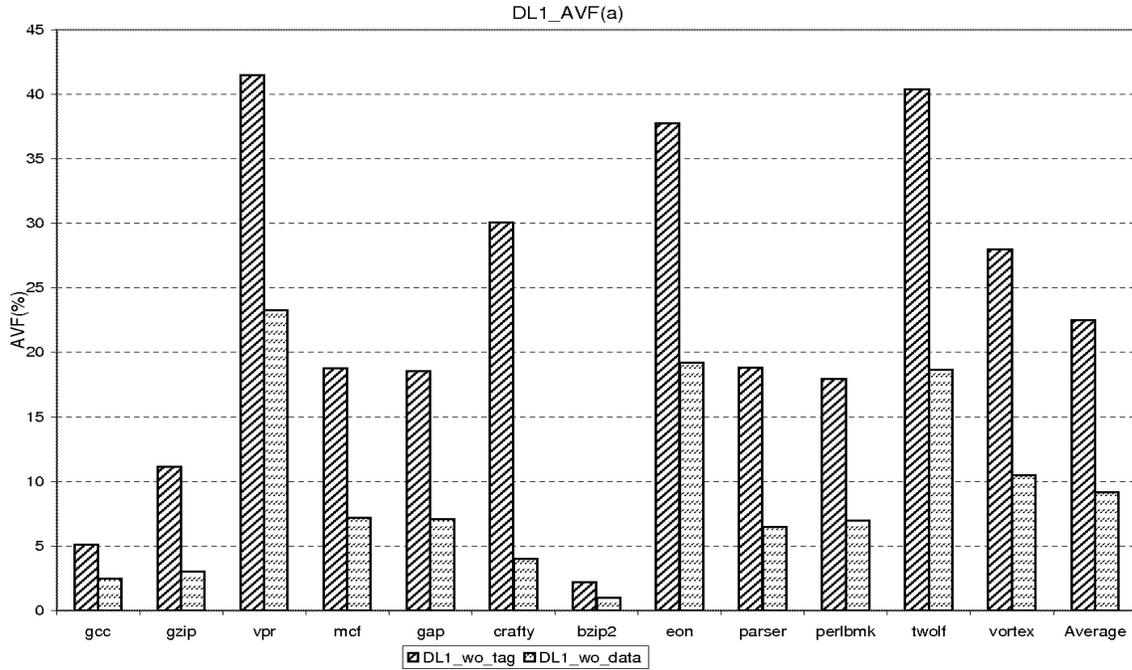


Figure 5.20: AVF of Level 1 Data Cache with Modified Razor flip-flops.

AVF of the data cache reduces by 72.14% to a new value of 9.14%.

Figures 5.22 and 5.23 show the reduced average AVFs for the Load Queue and the Store Queue. Figures 5.24 and 5.25 show the percentage reduction in the average AVF of the respective structures after Modified Razor flip-flops are introduced. We see that for both the structures, we obtain a maximum reduction (56.73% reduction for LQ and 55.77% reduction for SQ) in the average AVF when our Modified Razor flip-flops are introduced in the address fields of the structures. The reduced average AVFs are 8.98% and 13.43% for the LQ and the SQ respectively.

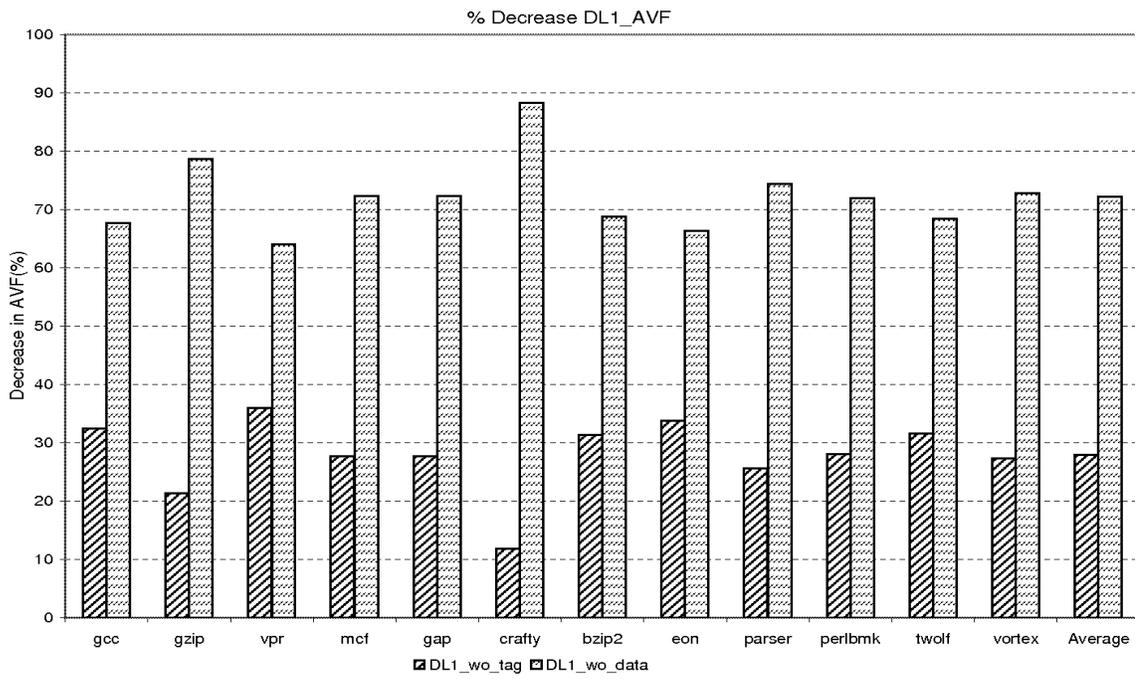


Figure 5.21: Percentage decrease in AVF of Level 1 Data Cache with Modified Razor flip-flops.

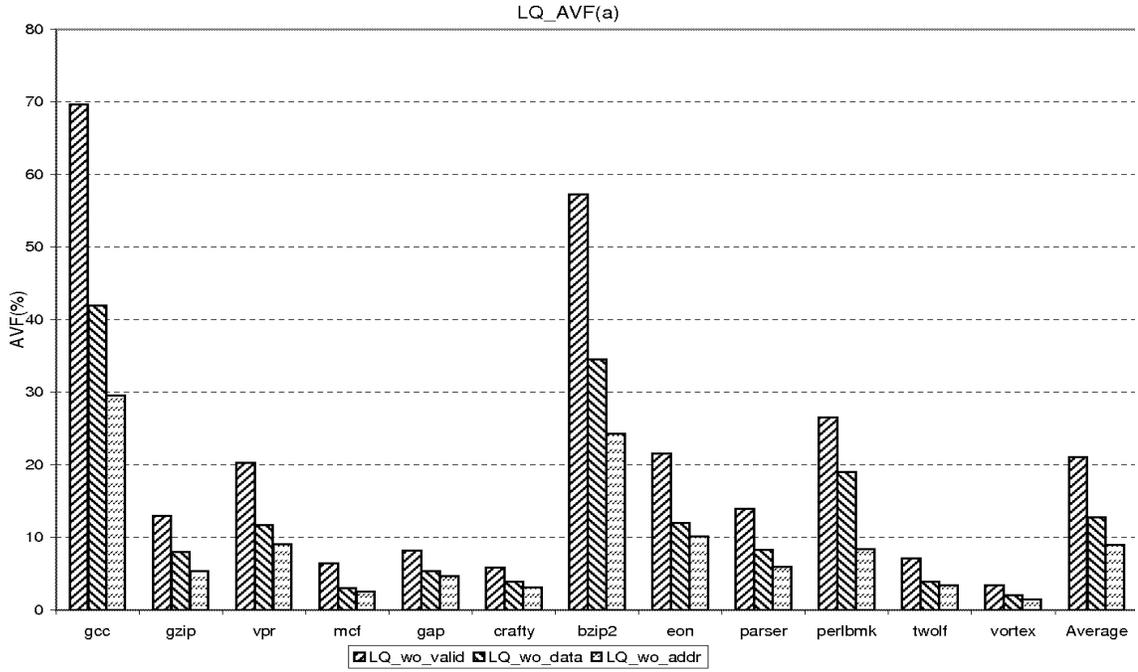


Figure 5.22: AVF of Load Queue with Modified Razor flip-flops.

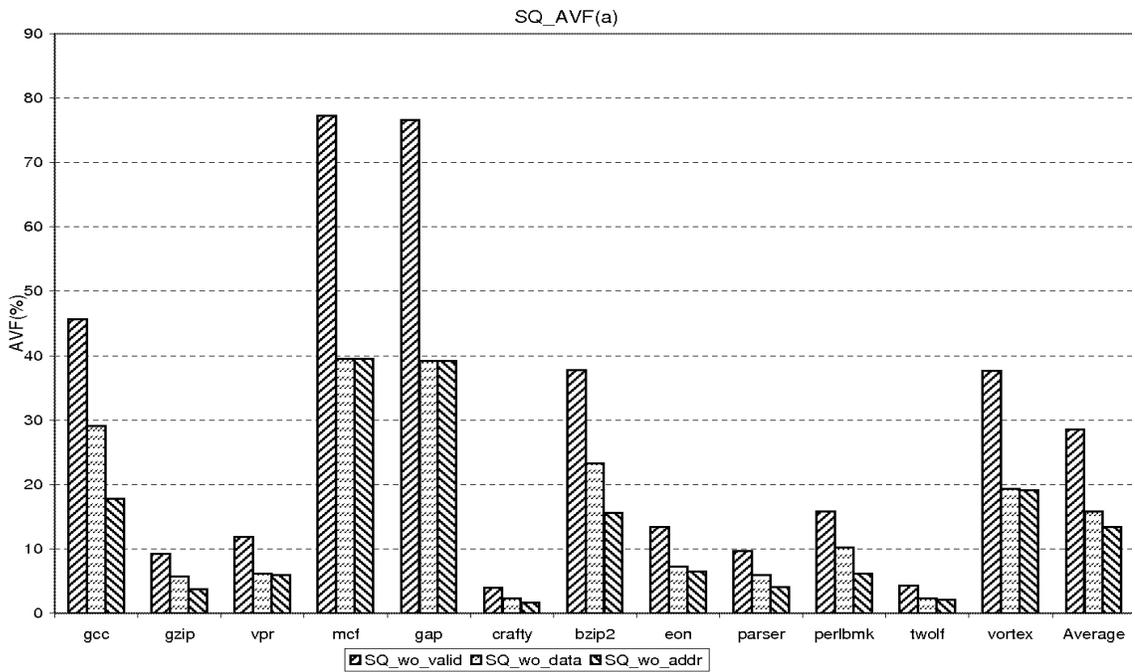


Figure 5.23: AVF of Store Queue with Modified Razor flip-flops.

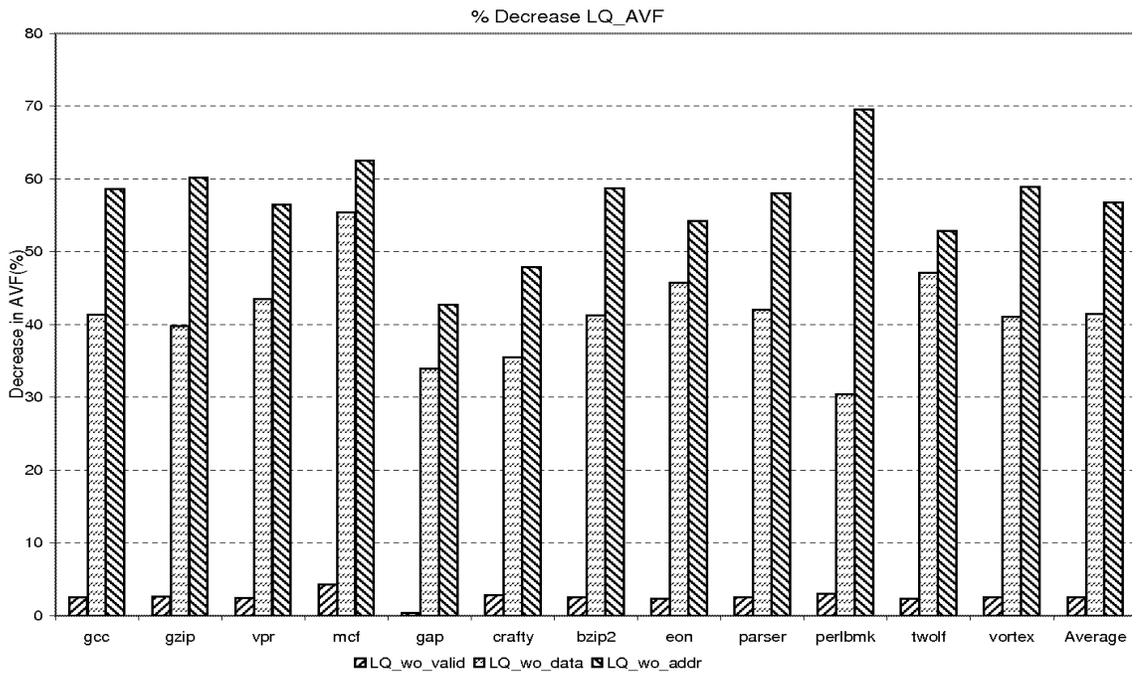


Figure 5.24: Percentage decrease in AVF of Load Queue with Modified Razor flip-flops.

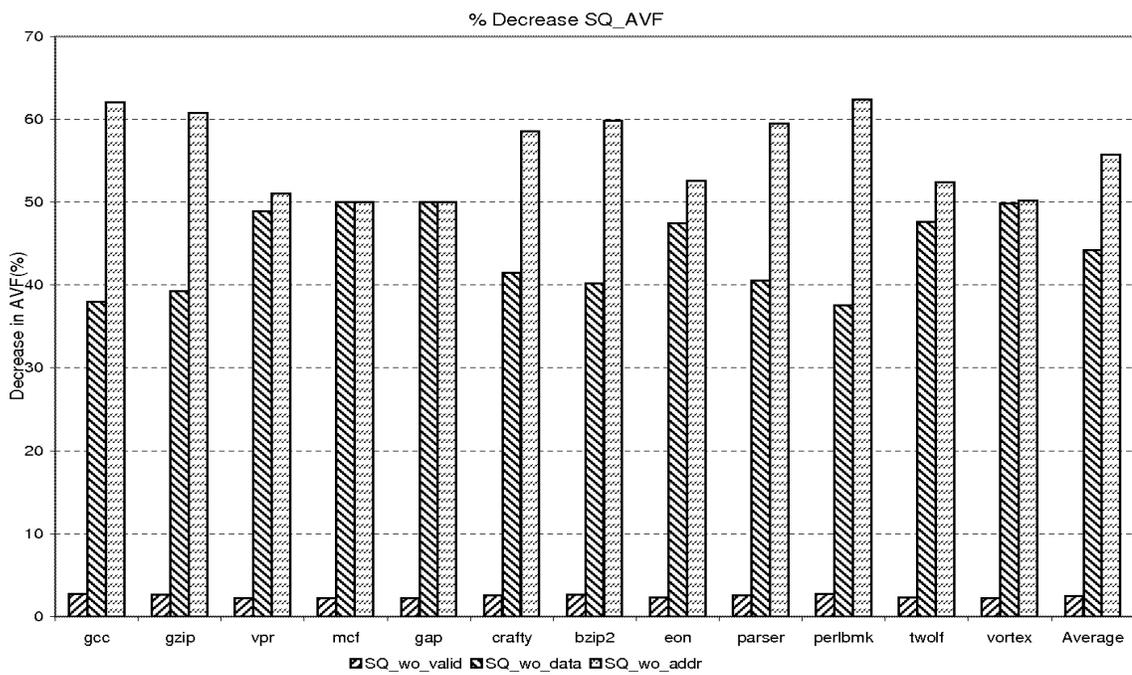


Figure 5.25: Percentage decrease in AVF of Store Queue with Modified Razor flip-flops.

5.5 Increase in Area and Power

Introduction of Modified Razor flip-flops also cause a decrease in the AVF of the microarchitectural structure and also the overall processor. However, introduction Modified Razor flip-flops cause an increase in the area requirement and power consumption for the microarchitectural structures because of reasons explained in Chapter 4. We have modeled the microarchitectural elements in RTL to measure the exact increase in area and power by introducing Modified Razor flip-flops in some fields of the microarchitectural elements viz. IQ, ROB, LQ and SQ. The area and power estimates for the Data Cache are obtained using Cacti [14]. Figures 5.26, 5.27, 5.28, and 5.29 show the area and power estimates for each of the microarchitectural structure under consideration. We have measured the area requirement and power consumption by introducing Modified Razor flip-flops in different fields which results in reduction in AVF. In addition to the fields identified in previous section, we have estimated the increase in area and power consumption by introducing Modified Razor flip-flops in all the fields of the microarchitectural structures under consideration. This would result in an 100% reduction of AVF but we observed that area and power increase by more than 100%.

From Figures 5.26(a), 5.27(a), and 5.29(a), we see that the area requirement for both combinational and sequential elements increase. The area requirement for sequential elements is due to the redundant flip-flop present inside our Modified Razor flip-flop which facilitates error detection. The increase in the combinational elements is due to the introduction of cells that implement the XOR comparators

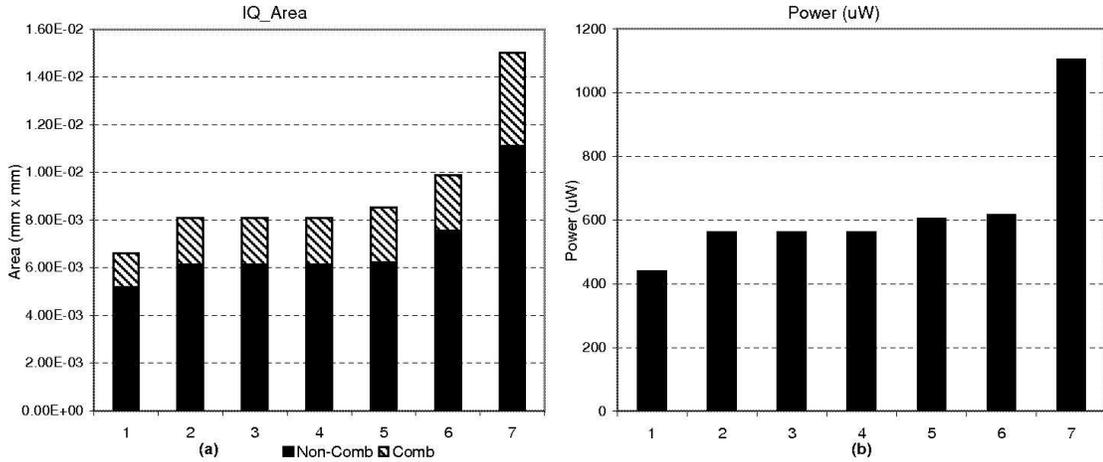


Figure 5.26: (a)Area and (b)Power consumption of Instruction Queue with Modified Razor flip-flops.

Legend - 1:IQ, 2:IQ_wo_operand1, 3:IQ_wo_operand2, 4:IQ_wo_dest, 5:IQ_wo_opcode, 6:IQ_wo_opcode_dest, 7:IQ_full

and also the ORing of the individual Error_L signals from each of the Modified Razor flip-flops.

5.6 Cost effective Percentage decrease in AVF

The main aim of the thesis is to find the most optimal choice for introducing Modified Razor flip-flops in microarchitectural elements. We saw that AVF reduces with the introduction of Modified Razor flip-flops but the area and power requirements increase which in-turn results in an increase in cost. Consider the following scenario - From Figures 5.20 and 5.21, we saw that when Modified Razor flip-flops are introduced in the data array of the Data Cache, we get an average reduction of 72.14%. From our estimates in Figure 5.28, we see that the area requirement and power consumption for this case goes up significantly to levels which are not

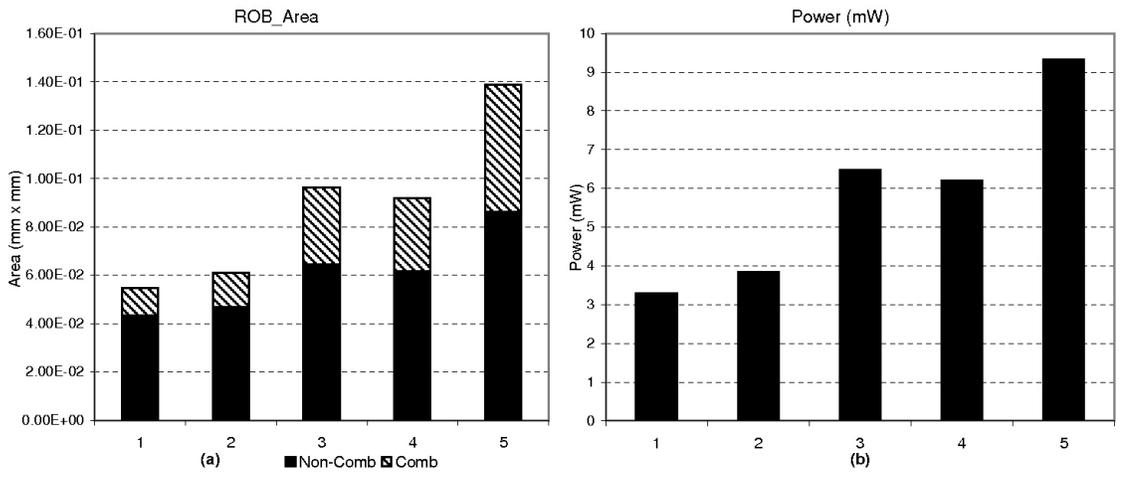


Figure 5.27: (a)Area and (b)Power consumption of Reorder Buffer with Modified Razor flip-flops.

Legend - 1:ROB, 2:ROB_wo_destopr, 3:ROB_wo_destopr_oprval, 4:ROB_wo_PC, 5:ROB_full

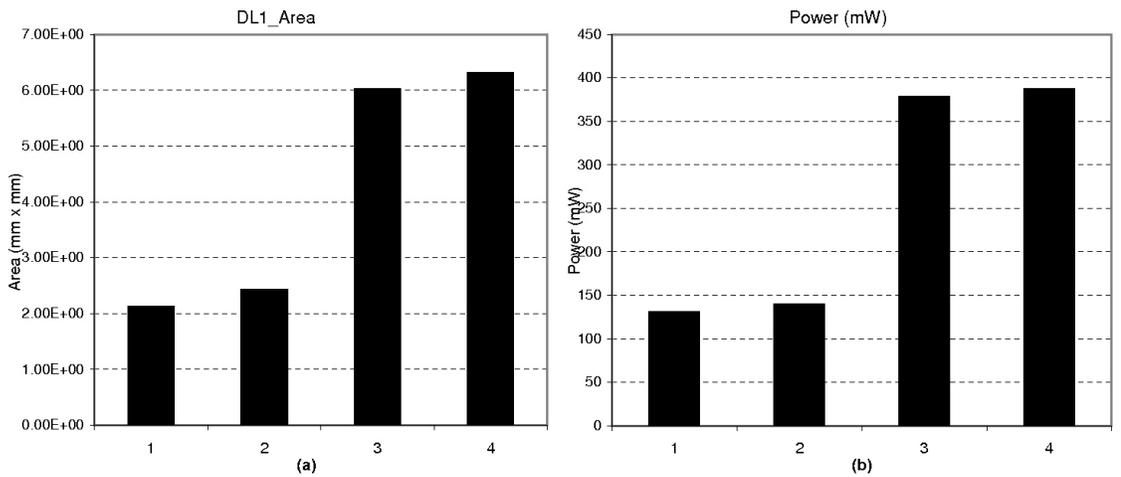


Figure 5.28: (a)Area and (b)Power consumption of Level 1 Data Cache with Modified Razor flip-flops.

Legend - 1:DL, 2:DL_wo_tag, 3:DL_wo_data, 4:DL_full

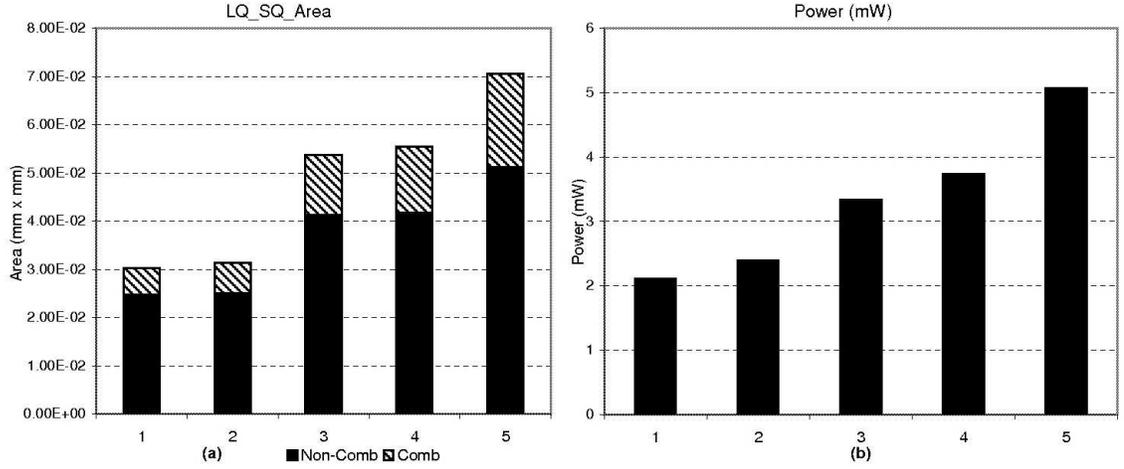


Figure 5.29: (a)Area and (b)Power consumption of Load/Store Queue with Modified Razor flip-flops.

Legend - 1:LQSQ, 2:LQSQ_wo_valid, 3:LQSQ_wo_addr, 4:LQSQ_wo_data, 5:LQSQ_full

practical (183.4% increase in area and 189% increase in power). To obtain the most cost effective choices of fields for the Modified Razor flip-flops, we have measured the Percentage decrease in AVF per Area-Power product. Figures 5.30, 5.31, 5.32, 5.33, 5.34 show the percentage decrease in per $mm^2 - mW$ for the microarchitectural elements considered in this thesis. 5.7 shows the most optimal choice of fields in the microarchitectural structures where the introduction Modified Razor flip-flops results in a most cost effective reduction in AVF.

5.7 Observations

Introduction of Modified Razor flip-flops causes a reduction in the AVF of the microarchitectural structures and hence the AVF of the processor. However, this also causes an increase in cost factor and more specifically the area requirement and

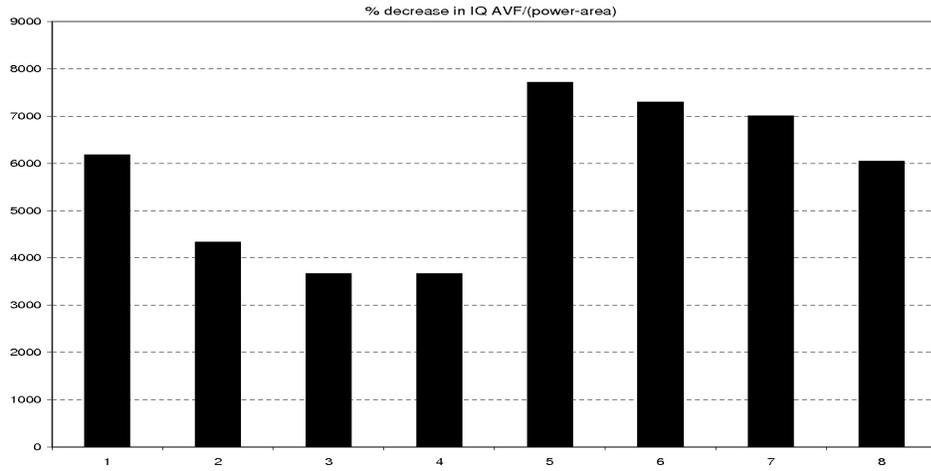


Figure 5.30: Percentage decrease in AVF per Area-Power product of Instruction Queue with Modified Razor flip-flops.

Legend - 1:IQ_wo_opcode, 2:IQ_wo_dest, 3:IQ_wo_operand1, 4:IQ_wo_operand2, 5:IQ_wo_opcode_dest, 6:IQ_wo_opcode_opr1, 7:IQ_wo_opcode_opr2, 8:IQ_full

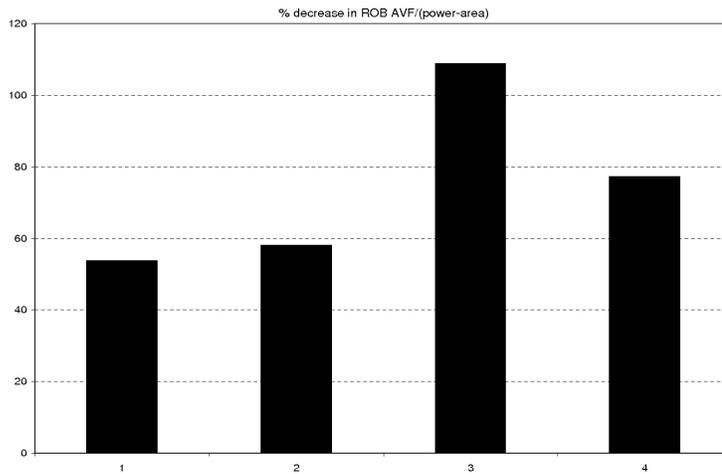


Figure 5.31: Percentage decrease in AVF per Area-Power product of Reorder Buffer with Modified Razor flip-flops.

Legend - 1:ROB_wo_destopr, 2:ROB_wo_destopr_oprval, 3:ROB_wo_PC, 4:ROB_full

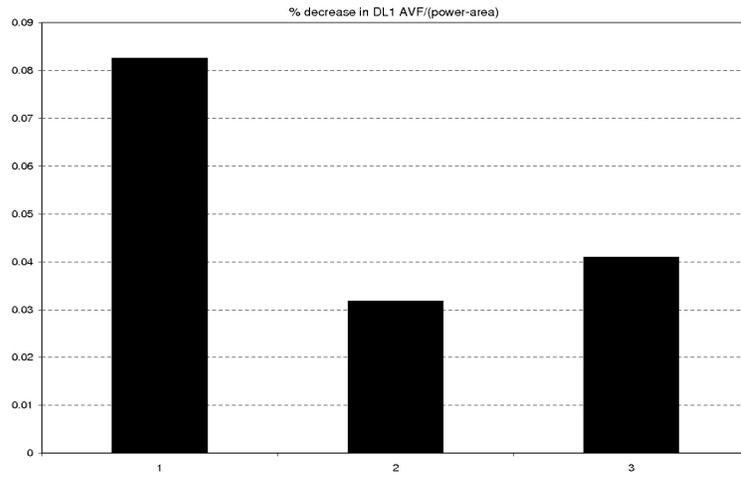


Figure 5.32: Percentage decrease in AVF per Area-Power product of Level 1 Data Cache with Modified Razor flip-flops.
 Legend - 1:DL_wo_tag, 2:DL_wo_data, 3:DL_full

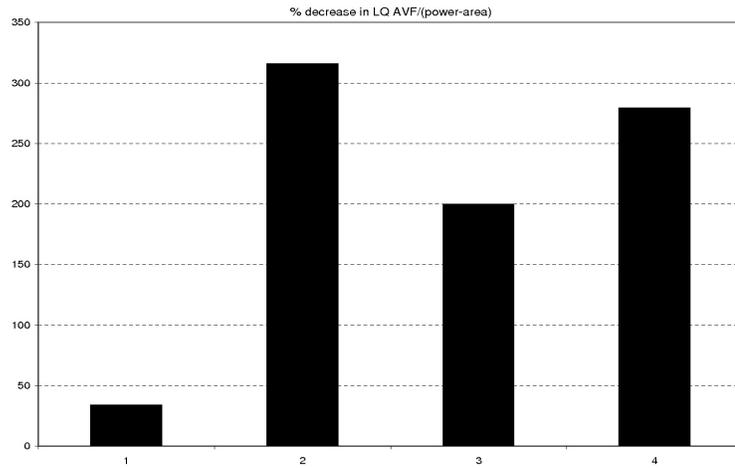


Figure 5.33: Percentage decrease in AVF per Area-Power product of Load Queue with Modified Razor flip-flops.
 Legend - 1:LQ_wo_valid, 2:LQ_wo_addr, 3:LQ_wo_data, 4:LQ_full

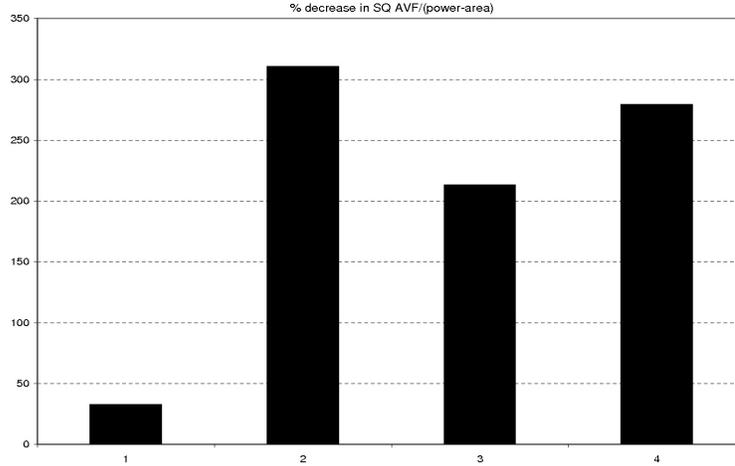


Figure 5.34: Percentage decrease in AVF per Area-Power product of Store Queue with Modified Razor flip-flops.

Legend - 1:SQ_wo_valid, 2:SQ_wo_addr, 3:SQ_wo_data, 4:SQ_full

the power consumption. We found from our experiments that, with the introduction of our Modified Razor flip-flops in certain fields of the microarchitectural structures, the most cost effective decrease in AVF can be achieved. The overall AVF before the introduction of all the microarchitectural elements is 22.5%. With the introduction of Modified flip-flops in fields shown in Table 5.7 the overall AVF reduces to 15.27%, which gives a reduction of 32.23%. The overall increase in area is 16.99% and the overall increase in power is 10.33%.

Microarchitectural Structure	Field for Modified Razor flip-flop
Instruction Queue	Opcode Destination Register Specifer
Reorder Buffer	Program Counter Field
Level 1 Data Cache	Tag array
Load Queue	Source Address field
Store Queue	Destination Address field

Table 5.3: Fields for Modified Razor flip-flops.

Chapter 6

Summary and Conclusion

6.1 Summary

Soft errors have become a key challenge in modern microprocessor design. Single Event Upsets and transient faults are a major source of soft errors. Eventhough the soft error rates of individual transistors are projected to remain as roughly the same for the next several technology generations, the overall per-chip fault rates will continue to increase exponentially in accordance to Moore's law. As a result, even logic elements, which were not a great concern in the reliability perspective earlier, have become a major source of concern. In this thesis, we address the issue of ensuring reliability in the operation of the microarchitectural elements in a cost effective fashion.

Soft errors that cause errors can be broadly classified as those which cause errors in program outcomes and those which do not. Errors that do not cause errors in program outcome do not pose a threat to the reliability of the processor. The reliability of the processor and its microarchitectural structures are measured in terms of Architecture Vulnerability Factor (AVF). AVF is defined as the probability that a fault in a processor structure will result in a visible error in the final output of a program. In this thesis, we have focused our effort on performing a detailed analysis of AVF of the microarchitectural elements viz. the Instruction Queue (IQ),

the Reorder Buffer (ROB), the Level 1 Data Cache (DL1), the Load Queue (LQ), and the Store Queue (SQ). We have identified the fields of these structures which contribute most significantly towards the overall AVF.

The technique to decrease the AVF of a structure is to ensure program correctness even with single bit soft errors. In this thesis, we have made design changes to incorporate error detection capabilities into Razor flip-flops and introduced them into one or more fields of structures that have significant contribution towards AVF. We have measured the resulting decrease in the AVF and also the increase in area requirement and power consumption. In order to find the most cost effective solution, we have measured the percentage decrease in AVF per area-power product for all the combinations of fields with Modified Razor flip-flops. We then find the most optimal solution and measure the overall decrease in AVF and increase in area and power.

6.2 Conclusion

In this thesis, we have proposed an approach to maintain program correctness by using the Modified Razor flip-flops in selected fields of the microarchitectural structures viz. the IQ, the ROB, the DL1, the LQ, and the SQ. Since this caused an increase in area and power, we identify the most optimal choices of fields which provide the highest percentage decrease in AVF per the area-power product. From our work, the fields are identified as Opcode and Destination Operand fields of IQ, Program Counter fields of ROB, Tag array of the DL1 and Address fields of the

LQ and the SQ. We found that with Modified flip-flops in the mentioned fields, the AVF decreases by 32.23%. The area increases by 16.99% and the power consumption increases by 10.33%.

Bibliography

- [1] H. Nguyen and Y. Yagil, "A systematic approach to ser estimation and solutions," in *Proceedings of the 41st annual IEEE International Symposium on Reliability Physics*, (Texas, USA), pp. 60–70, IEEE, 2003.
- [2] J. C. Smolens, B. T. Gold, J. Kim, B. Falsafi, J. C. Hoe, and A. G. Nowatzky, "Fingerprinting: Bounding soft-error-detection latency and bandwidth," *IEEE Micro*, vol. 24, no. 6, pp. 22–29, 2004.
- [3] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, (Washington, DC, USA), p. 7, IEEE Computer Society, 2003.
- [4] R. Baumann, "Soft errors in commercial semiconductor technology: Overview and scaling trends," *IEEE 2002 Reliability Physics Tutorial Notes, Reliability Fundamentals*, pp. 1–14, December 2003.
- [5] L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Soft error and energy consumption interactions: a data cache perspective," in *Proceedings of the International Symposium on Low Power Electronics and Design*, (New York, NY, USA), pp. 132–137, ACM, 2004.
- [6] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, (Washington, DC, USA), p. 29, IEEE Computer Society, 2003.
- [7] A. Biswas, P. Racunas, R. Cheveresan, J. Emer, S. S. Mukherjee, and R. Rangan, "Computing architectural vulnerability factors for address-based structures," in *Proceedings of the 32nd annual international symposium on Computer Architecture*, (Washington, DC, USA), pp. 532 – 543, IEEE Computer Society, 2005.
- [8] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron cmos technology," *IEEE Transactions on Nuclear Science*, vol. 43, pp. 2874–2878, December 1996.
- [9] H. Ando, Y. Yoshida, A. Inoue, Itsumi Sugiyama, T. Asakawa, K. Morita, T. Muta, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita, and H. Sugiyama, "A 1.3gtiz fifth generation sparc64 microprocessor," in *Proceedings of the Design Automation Conference*, pp. 702–705, June 2003.

- [10] H. Kim, H.-J. Jeon, K. Lee, and H. Lee, "The design and evaluation of all voting triple modular redundancy system," in *Proceedings of the Reliability and Maintainability Symposium*, (Seattle, WA, USA), pp. 439 – 444, IEEE, 2002.
- [11] T. J. Slegel, R. M. A. III, M. A. Check, B. C. Giamei, B. W. Krumm, C. A. Krygowski, W. H. Li, J. S. Liptay, J. D. MacDougall, T. J. McPherson, J. A. Navarro, E. M. Schwarz, K. Shum, and C. F. Webb, "Ibm's s/390 g5 microprocessor design," *IEEE Micro*, vol. 19, no. 2, pp. 12–23, 1999.
- [12] T. Austin, "Diva: A dynamic approach to microprocessor verification," *Journal of Instruction Level Parallelism 2*, vol. 1, May 2000.
- [13] S. Chatterjee, C. Weaver, and T. Austin, "Efficient checker processor design," in *MICRO 33: Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, (New York, NY, USA), pp. 87–97, ACM, 2000.
- [14] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model," WRL Research Report 2001/2, Compaq Western Research Laboratory, Palo Alto, California 94301 USA, August 2001.
- [15] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, and B. Chin, "Ibm experiments in soft fails in computer electronics (1978-1994)," *IBM J. Res. Dev.*, vol. 40, no. 1, pp. 3–18, 1996.
- [16] E. Normand, "Single event upset at ground level," *IEEE Transactions on Nuclear Science*, vol. 43, pp. 2742–2750, December 1996.
- [17] W. Bryg and J. Alabado, "The ultrasparc t1 processor - reliability, availability, and serviceability," *Whitepapers: UltraSPARC Processors Documentation*, December 2005.
- [18] T. Karnik, B. Bloechel, K. Soumyanath, G. Dermer, V. De, and S. Borkar, "Scaling trends of cosmic rays induced soft errors in static latches beyond 0.18μ ," *Symposium on VLSI Circuits Digest of Technical Papers*, pp. 61–62, 2001.
- [19] T. Calin, M. Nicolaidis, and R. Velazco, "Impact of scaling on soft-error rates in commercial microprocessors," *IEEE Transactions on Nuclear Science*, vol. 49, pp. 3100–3106, December 2002.
- [20] X. Fu, T. Li, and J. Fortes, "Sim-soda: A unified framework for architectural level software reliability analysis," *Workshop on Modeling, Benchmarking and Simulation*, June 2006.
- [21] R. Desikan, D. Burger, S. Keckler, and T. Austin, "Sim-alpha: a validated execution driven alpha 21264 simulator," *Technical Report TR-01-23, Department of Computer Sciences, University of Texas at Austin.*, 2001.

- [22] C. Corporation, “Alpha 21264/ev67 microprocessor hardware reference manual,” September 2000.
- [23] J. L. Henning, “Spec cpu2000: Measuring cpu performance in the new millennium,” *IEEE Computer*, vol. 33, no. 7, pp. 28–35, 2000.
- [24] R. Desikan, D. Burger, and S. Keckler, “Experimental error in microprocessor simulation,” in *Proceedings of the 28th annual IEEE international symposium on Computer Architecture*, (Los Alamitos, CA, USA), pp. 266–277, IEEE Computer Society Press, 2001.
- [25] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically characterizing large scale program behavior,” in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, (New York, NY, USA), pp. 45–57, ACM, 2002.
- [26] P. Kurup and T. Abbasi, “Logic synthesis using synopsys,” 1997.
- [27] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel, “Characterizing the Effects of Transient Faults on a High-Performance Processor Pipeline,” in *Proceedings of the International Conference on Dependable Systems and Networks*, 2004.
- [28] C. Weaver, J. Emer, S. S. Mukherjee, and S. K. Reinhardt, “Techniques to Reduce the Soft Error Rate of a High-Performance Microprocessor,” in *Proceedings of the International Symposium on Computer Architecture*, 2004.
- [29] T. Karnik, and P. Hazucha, “Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes,” in *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. :128–143, June 2004.
- [30] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, “Robust System Design with Built-In Soft-Error Resilience,” in *IEEE Computer*, vol. 38, no. 2, pp. :43–52, Feb. 2005.
- [31] S. Hareland, J. Maiz, M. Alavi, K. Mistry, S. Walstra, and C. Dai, “Impact of CMOS Scaling and SOI on soft error rates of logic processes,” in *VLSI Technology Digest of Technical Papers*, 2001.