NAIST-IS-DD1161024

Doctoral Dissertation

Studies on Improving reliability and efficiency using Adaptive redundancy in LSI system

Yukihiro Sasagawa

March 15, 2013

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Yukihiro Sasagawa

Thesis Committee:

Professor Yasuhiko Nakashima	(Supervisor)
Professor Michiko Inoue	(Co-supervisor)
Assistant Professor Jun Yao	(Co-supervisor)

Studies on Improving reliability and efficiency using Adaptive redundancy in LSI system*

Yukihiro Sasagawa

Abstract

Recently, the DVS (Dynamic Voltage Scaling) method has been aggressively applied to processors with Razor Flip-Flops. With Razor FF detecting setup errors, the supply voltage in these processors is downscaled to a near critical setup timing level for a maximum power consumption reduction. However, the conventional Razor and DVS combinations cannot tolerate well error rate variations caused by IR-drops and environment changes. At the near critical setup timing point, even a small error rate change will result in a sharp performance degradation. This dissertation proposes RazorProtector, a DVS application method based on a redundant data-path which uses a multi-cycle redundant calculation to shorten the recovery penalty after a setup error occurrence.

A dynamic redundancy-adapting scheme is also given to use the designed redundant data-path effectively based on a study of the program, device and error rate characteristics. This dissertation employs a metric DCF (Delay Criticality Factor) to measure the sensitivity to the setup error of each calculation type. The results show that RazorProtector with the adaptive redundancy architecture can, compared to the traditional DVS method with Razor FF, under a large setup rate caused by a 10% unwanted voltage drop, reduce Energy Delay Product (EDP) up to 61% at 100μs/V, 85% at 200μs/V voltage scaling slope.

This dissertation also applied a program characteristic-based tuning method dynamically to adapt the redundancy level of RazorProtector. The most related

* Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1161024, March 15, 2013.

parameters, including ILP (Instruction Level Parallelism) and DCF are selected and effectively reflected in the tuning to give an adaptive redundancy dynamically. The simulation results show that the adaptive redundancy can achieve better EDP reduction than any static controls under a workload suite with different behaviors. Compared to the original DVS method, the proposed dynamic control achieves an average 56% EDP reduction for the interested workloads in benchmark applications.

Keywords:

adaptive redundancy, setup error recovery, DVS, low power, AVF

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For the past years, the process scaling has been providing an exponential increase of per die transistor numbers, which leads to a major contribution for the performance-oriented design of electronic devices. This rapid scaling speed is expected to continue in a near future, as reported in ITRS-2011 [1], which is also given in Fig. 1.1. According to the data in the figure, there will be a 9x increase in the transistor density from year 2012 to 2022. However, with the increasing concern for battery life and thermal design problems, simply going toward performance will gradually shift to the recent energy efficient-oriented target from the market requirements. The data of power consumption per area in Fig. 1.1 and the data of required power per million transistors in Fig. 1.2 also show that the per transistor power consumption will be reduced along the technology scaling. This mainly comes from the ultimate voltage miniaturization along the technology scaling. However, with a lowered working voltage, the future transistors are likely to be more vulnerable to the sudden IR-drops. Design margins are thereby required to eliminate the possible timing faults consequently, though this is on the opposite direction of power reductions.

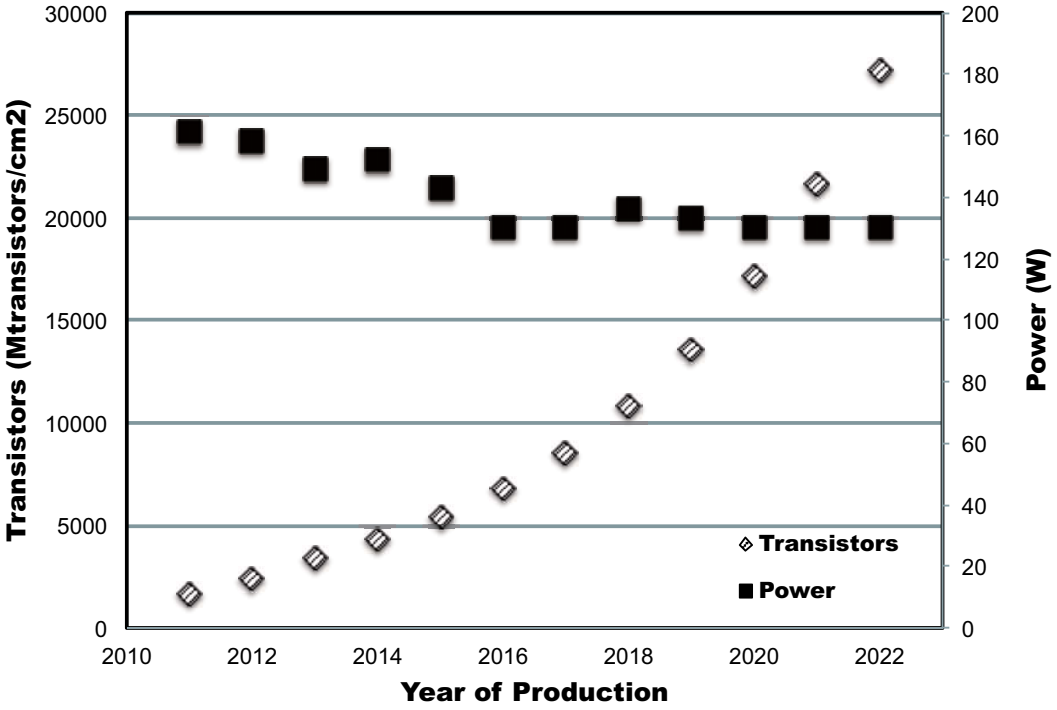


Figure 1.1. Trend of transistors and power reported in ITRS-2011.

Many technologies [2, 3, 4, 7, 8] have been applied to eliminate unnecessary design margins and reduce resources or supply voltages in order to effectively fulfill both the calculation correctness and a limited power budget requirements. Among these technologies, one recent major technique is to use Razor Flip-Flop to help aggressively apply the well-known low power method DVS (Dynamic Voltage Scaling), as is introduced in papers [6, 9, 10, 11]. Razor FF uses a set of primary/shadow latches in the critical computation paths to detect setup errors caused by voltage over-scaling. It is thus possible to lower the voltage to a near critical setup timing level. A maximum reduction of power consumption can be achieved when the cost of setup error recovery and the gain from voltage downscaling reach a balancing point.

A major constraint of traditional DVS application in a processor with Razor

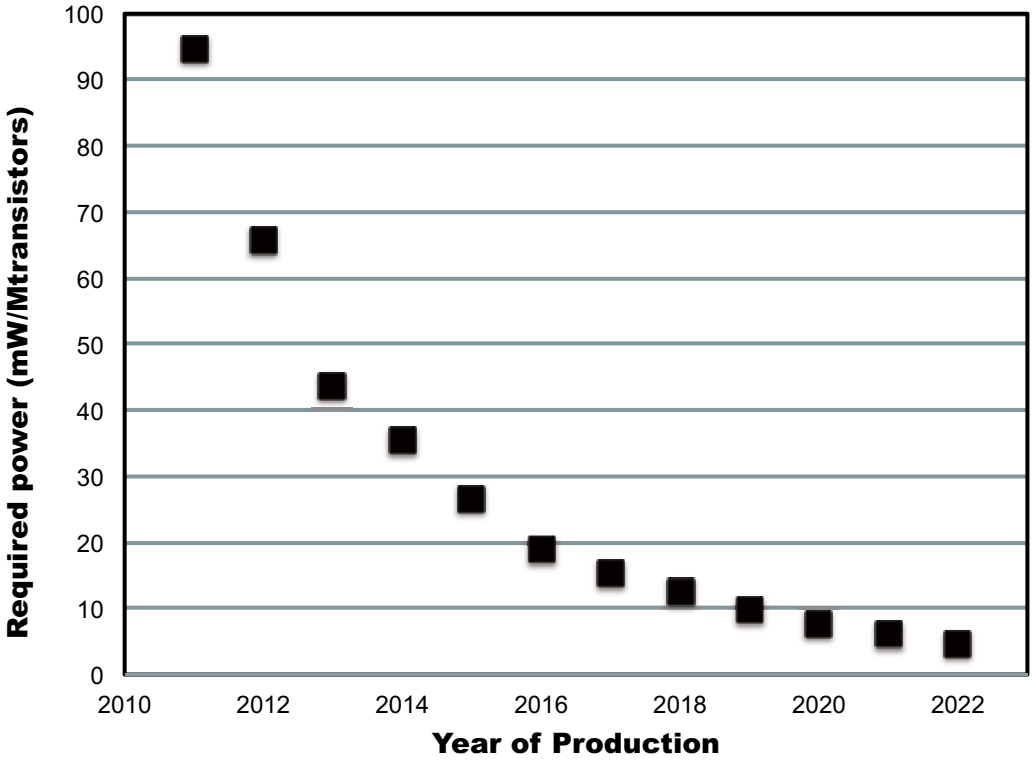


Figure 1.2. Estimated power requirement based on ITRS-2011.

FF support comes from the relatively long recovery penalty when Razor FF detects a setup error. A pipeline flush is required to restore the processor status back to the correct execution route. In large IR-drop zones, frequent pipeline flushes will give a sharp decrease of the performance so that a voltage re-scaling is required to reach another balanced working point. However, this voltage re-scaling will additionally come with a microsecond order penalty. Some recent researches [22, 23] have indicated that it is possible to use additional power supply networks or an on-chip DC-DC converter to allow a nanosecond level voltage scaling, which can thus alleviate the voltage scaling delay penalty in Razor-FF processors. However, it is still impractical and not used in commercial processors due to the instability of these power supplies. In addition, these fast voltage

scaling techniques will complicate the power/clock domains and add to LSI design restriction. Therefore, this research still uses microseconds order voltage scaling, following major modern technologies [24]. Instead, this research explored architectural approaches to cover this voltage scaling penalty.

This dissertation proposes RazorProtector that uses an architectural redundant path as an alternative so as to help avoid the sharp performance degradation in large IR-drop zones. The redundant path works on the same instruction stream as the normal path. However, multi-cycle calculations are used in the redundant path, which guarantees a setup-error-free execution. Under a frequent timing error occurrence that is usually near the balancing point, the redundant path can help provide a very fast recovery. With this fast recovery scheme, it is possible to tolerate more IR-drops before a new balancing point is required. This method thus avoids some frequent voltage scalings and contributes to provide more energy reductions especially under large IR-drops.

The remainder of this dissertation is organized as follows. Chapter 2 introduces related works and raises the challenges for each work. Chapter 3 explains the targets of this research. Chapter 4 presents the basic structure of the RazorProtector method as a solution to the above problems. Its working scheme to use redundant data-path to provide a fast setup error recovery, and the comparison result to a traditional method will be introduced in detail. This chapter also gives the redundancy-adapting algorithm by using the metric DCF, and shows the benefits of this method based on a mathematical model. Chapter 5 further presents the method in Chapter 4 by adding dynamic adaptation of program characteristics in the application of the RazorProtector method. The major control parameter, as the $\mathrm{RISK_{th}}$, is tuned to best fit the periodical information inside the running processor. Chapter 6 concludes the whole dissertation.

Chapter 2

Related Work

Dynamic voltage scaling (DVS) is a well-used method to lower the power consumption in electronic devices as it provides a quadric effect in reducing the dynamic power and a linear effect in lowering the static power. Many power management methods are composed of various monitoring technology based on this voltage scaling. This chapter will give a fast survey of the power saving method especially make use of voltage scaling.

Intel SpeedStep[2] is a well-known DVS technology in commercial products. This approach presumes sets of frequency and voltage combinations based on design specification and these settings will be dynamically used according to runtime monitoring in various systems [6]. As the settings of different voltage and frequency pairs are determined at the processor design phase, it will be referred as *Design-time DVS* in this dissertation. However, due to the static bindings between voltage and frequency, *Design-time DVS* is not able to cover the variation of data dependencies, process variations and condition changes.

Methods to measure the runtime logic delay are used to cover the process variations and condition changes, as a preparation to reduce the working power more accurately. Papers [7, 4] proposes to implement a sample circuit, whose delay is determined to be a representative one of the whole LSI. As an alternative, the paper [3] proposes a monitoring circuit based on triple latches to detect the delay of the specific part of the LSI. Both the sample circuit in papers [7, 4] and the triple-latch monitor work under an off-line mode to the real program execution, which can provide an effective measure of the processor variations.

However, due to their off-line working feature, these two methods are still not able to detect timing faults that mainly come from data dependency between long data-paths and unexpected dynamic condition changes like IR-drops.

The setup faults can be detected by redundancy based circuits like setup error alerting flip-flop (Canary FF) [8]. The Canary FF is designed to be triggered by a delayed clock, and is added into the system beside a main flip-flop. The comparison between the main FF and the canary FF gives an error prediction. In its application, the delay line used to trigger the canary FF contains the margins to guarantee that the alert signal (error prediction) must happen in advance of a real setup error. Thus, the power management itself implies design margins to avoid unexpected malfunction. It had been dilemma for various methods aiming to reduce design margins.

Above power management concepts focus on optimizing frequency and voltage, which can firstly guarantee correct functionality and secondly achieve a good energy reduction. In other words, the worst case voltage and frequency pair is still firstly fulfilled no matter how the normal cases will be. Compared to above approaches, Razor technology [9] provides a novel method which allows malfunctions but adds recovery scheme to solve such dilemma. Razor FF uses a set of primary/shadow latches in the critical computation paths to detect setup errors caused by voltage over-scaling. It is thus possible to lower the voltage to a near critical setup timing level. A maximum reduction of power consumption can be achieved when the cost of setup error recovery and the gain from voltage downscaling reach a balancing point. Since the first proposal of Razor had been raised, it can continuously help to reduce min-delay constraints and achieve low power consumption as in [10, 11]. Fig. 2.1 summarizes feature of various power management concepts.

However, Razor technology has an additional penalty in performance as it requires a recovery to restore the processor status to the correct route. Though each recovery may only take a very short architectural state flush inside the pipeline, it may still be visible under a possible large IR-drop zone, which still remains a challenge to explore the efficiency of Razor technique. It is also the major task of this research to provide solutions for this challenge.

Recently, a new approach improves efficiency from Razor by predicting error

and inserting an execution bubble speculatively [14]. Inserting an execution bubble creates enough time to avoid setup errors. It relies on speculative behavior and thus runs the risk of a large penalty as same as Razor. Fig. 2.2 summarizes feature of various redundant methods.

Power management Concept	Approach	Effectiveness of margin reduction			Performance degradation
		Data dependency	Process variation	Condition change	
Design-time DVS	Control by referring Frequency/Voltage database based on pre-defined design specification.	N/A	N/A	N/A	N/A
Delay Line Speed Detector, Triple-Latch Monitor	Control by measured actual logic delay based on sample circuits.	N/A	Effective, but need margin for sample circuit	Effective, but need margin for sample circuit	N/A
Canary-FF	Control by referring In-Situ setup-alert based on canary-FF.	Effective	Effective, but need margin for alert against error	Effective, but need margin for alert against error	N/A
Razor	Control by referring In-Situ setup-error rate based on error detecting FF.	Effective	Effective	Effective	Large because of retrying until V/f recover
Proposal in this dissertation		Effective	Effective	Effective	Small because of less retrying by redundancy

Figure 2.1. Comparison of various power management concepts.

Redundant method	Approach	Cost of redundancy	Performance degradation
Razor	Refer In-Situ information of setup error detecting FF (Razor-FF), and retry instruction.	Small because of simple detecting FF and retrying logic	Large because of retrying instruction
Speculatively bubble insertion	Predict setup error rate from setup error detecting FF, and insert bubbles of instruction speculatively.	Small because of simple detecting FF and inserting bubble logic	Large because of inserting bubbles
Proposal in this dissertation		Small because of simple detecting FF and reusing parallel data-path	Small because of less retrying by redundancy

Figure 2.2. Comparison of various redundant methods.

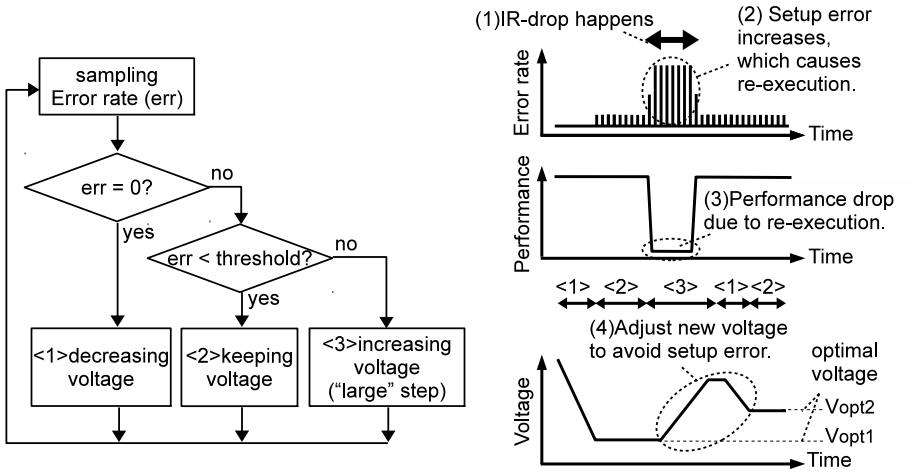
Chapter 3

Target of this research

As explained in previous chapter, Razor technology is a good approach to reduce design margins. However, there is penalty due to retrying in case of over-scaling.

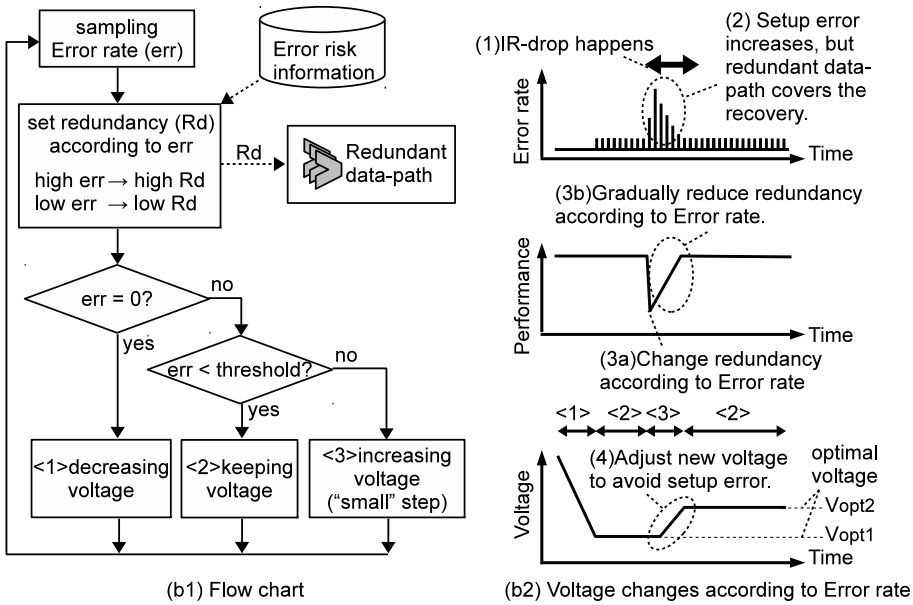
Fig. 3.1 (a) gives a voltage-adjusting scheme in a processor with conventional Razor-FF usage, described in the paper [11]. Specifically, steps $\langle 1 \rangle$ and $\langle 2 \rangle$ in Fig. 3.1 (a1) indicate the control flow before the balancing point V_{opt1} is reached. The 'err' here denotes the error rate detected by the Razor-FF. After step $\langle 2 \rangle$, it can be supposed that all the voltage margins have been removed and maximum power reduction has been achieved. The duration $\langle 1 \rangle$ and $\langle 2 \rangle$ in Fig. 3.1 (a2) show the corresponding timing error, performance and voltage in these two steps.

However, the near critical setup timing balance will have problems in facing sudden timing error rate variations caused by unexpected IR-drops and environment changes. Assume that there is a large timing error rate increase in duration $\langle 3 \rangle$ in Fig. 3.1. Accordingly, the processor performance will drop sharply since the previously balanced voltage is not sufficient to produce correct results. Computation units will keep retrying the same calculations. The processor is thus required to re-tune its balancing point, indicated as step $\langle 3 \rangle$, followed by steps $\langle 1 \rangle$ and $\langle 2 \rangle$ in Fig. 3.1 (a). As a large voltage change usually requires a long delay, which will be in the order of microseconds in modern processors, many cycles will be lost in step $\langle 1 \rangle$ before the new balancing point V_{opt2} is reached.



(a1) Flow chart (a2) Voltage changes according to Error rate

(a) Conventional DVS (e.g. using Razor)



(b1) Flow chart (b2) Voltage changes according to Error rate

(b) DVS with adaptive redundancy (proposal in this research)

Figure 3.1. Flow chart and voltage transition in DVS methods (Conventional and proposal).

Some recent researches [22, 23] has indicated that it is possible to use additional power supply networks or an on-chip DC-DC converter to allow a nanosecond level voltage scaling, which can thus alleviate the voltage scaling delay penalty in Razor-FF processors. However, it is still impractical and not used in commercial processors due to the instability of these power supplies. In addition, these fast voltage scaling techniques will complicate the power/clock domains and add to LSI design restriction. Therefore, this research still uses microseconds order voltage scaling, following major modern technologies [24]. Instead, this research explored architectural approaches to cover this voltage scaling penalty.

This dissertation proposes RazorProtector that uses an architectural redundant path as an alternative so as to help avoid the sharp performance degradation shown in Fig. 3.1 (a2). The redundant path works on the same instruction stream as the normal path. However, multi-cycle calculations are used in the redundant path, which guarantees a setup-error-free execution. Under a frequent timing error occurrence that is usually near the balancing point, the redundant path can help provide a very fast recovery.

The detailed control flow of this DVS application inside the redundant paths is shown in Fig. 3.1 (b). Before the tuning phases, as in $\langle 1 \rangle$ and $\langle 2 \rangle$ in Fig. 3.1 (b1), the execution redundancy will be determined and sent to the architecture control units. The need for a redundant execution arises from the combination of the current error rate, device delay feature, and the program flow. We employ a metric DCF (Delay Criticality Factor) to measure the sensitivity to the setup error of each calculation type. As seen in Fig. 3.1 (b2), the redundant path shows its benefits at the post balancing point regions. When a setup error rate variation occurs after V_{opt1} has been reached, the error-free redundant path can still provide a correct result, avoiding a sudden performance drop. It is thus possible to re-tune the balancing point by gradually stepping-up the supply voltage, shown as step $\langle 3 \rangle$ followed by $\langle 2 \rangle$ in Fig. 3.1 (b2). It is expected that the re-tuning phase can be accelerated by avoiding a large voltage increase, as in step $\langle 1 \rangle$ in Fig. 3.1 (a). Moreover, the redundant path can help maintain some throughput during the re-tuning phase.

Chapter 4

RazorProtector

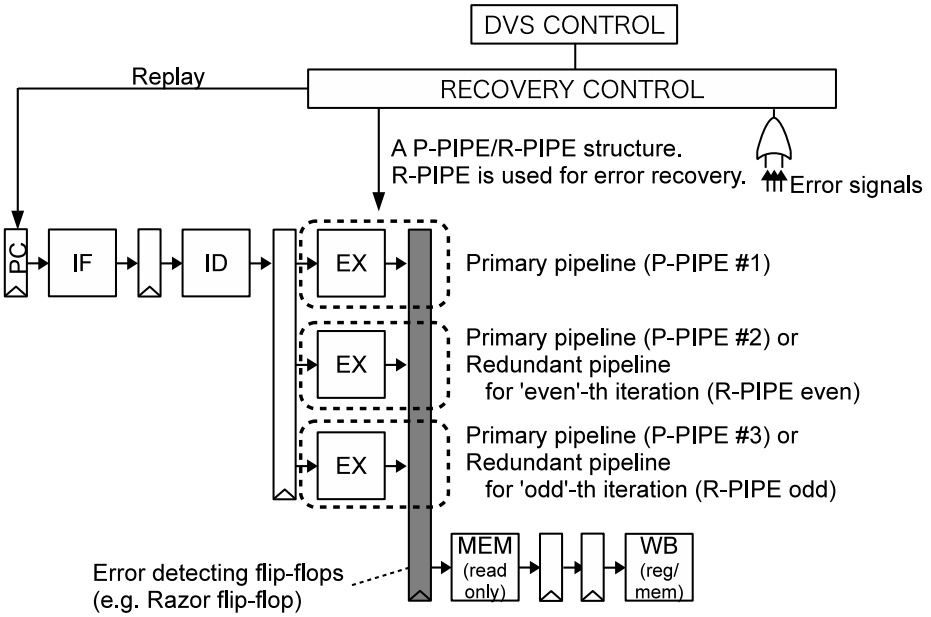
This chapter shows the idea of RazorProtector that uses an architectural redundant path. In Section 1, the basic structure of the redundant data-path based RazorProtector and its working scheme to provide a fast setup error recovery will be introduced. Section 2 gives the redundancy-adapting algorithm by using the metric DCF. Section 3 introduces the evaluation methodology. Section 4 and Section 5 present the energy and performance study of the RazorProtector, compared with the conventional DVS application in a Razor-FF processor. Section 6 concludes the whole of this chapter.

1. Redundant Data-Path

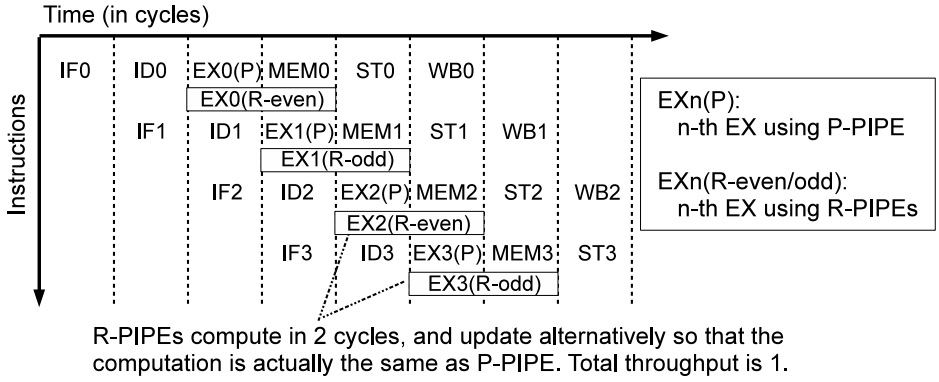
1.1 Basic Architecture

The redundant data-path structure is illustrated in Fig. 4.1 (a). The whole architecture is composed of three pipelines in total: one primary pipeline (P-PIPE) and two supplemental redundant pipelines (R-PIPEs). The P-PIPE works under a normal setting and produces a fast calculation which is, however, vulnerable to setup timing error. Under a Redundant mode, the R-PIPEs are designed to use two cycles to finish each calculation in order to alleviate the timing error pressure from the critical path, which is achieved by changing the clock speed of the latches connecting to the execution unit in the R-PIPEs. To detect timing errors under an aggressive voltage scaling, the pipeline registers follow the structure of

a Razor-FF [9, 10, 11]. An error signal indicated by any Razor-FF will trigger a setup error recovery.



(a) Structure of Redundant data-path in RazorProtector



(b) Pipeline execution for redundant mode (P-PIPE and 2 R-PIPEs)

Figure 4.1. Redundant data-path system.

Fig. 4.1 (b) illustrates an execution example of the triple-pipeline architecture under the Redundant mode. Four instructions from 0 to 3 will be executed twice in this redundant data-path structure. More specifically, when instructions with even numbers, such as 0 and 2, are issued to P-PIPE, their executions will also be started on the first R-PIPE. In the succeeding cycle, odd-numbered instructions will be issued to both P-PIPE and the second R-PIPE. Even when R-PIPE works under a halved throughput, the combination of two R-PIPEs can provide an additional execution of the four instructions. When the Razor-FFs in the P-PIPE detect no error, its result can be used safely and no recovery will be required. Regarding the consideration that an execution unit such as an ALU usually has a longer critical path and is more sensitive to IR-drops than units in other stages. For example, an Adder has critical paths correspond to carry propagation structure. The depth of each carry propagation path is different correspond to bit position. It means the chance of setup error depends on data itself even the operation/instruction is the same. On the other hand, control logics (e.g. Decoder) have delay correspond to operations/instructions. It is easy to keep correct functionality using traditional Razor technology rather than execution unit. This dissertation assumes that a redundant data-path only applies to EX stages.

As the two R-PIPEs use 2 cycles to complete the calculation and start execution in odd and even cycles respectively, they need to get the forwarded results from P-PIPE to finish the back-to-back dependent instruction series. As an example, instructions $i1 \rightarrow i2 \rightarrow i3$ are back-to-back dependent instructions, which correspond to EX1, EX2 and EX3 in Fig. 4.2. When R-PIPE (R-odd) works on $i3$, R-PIPE (R-even) is still working on the latter half execution of $i2$ so that it is not able to use the forwarding route from R-even to R-odd. However, as all of these three instructions are processed at the full speed of P-PIPE, we can use the $i2$ execution result of P-PIPE in R-odd, shown as the black arrows in Fig. 4.2.

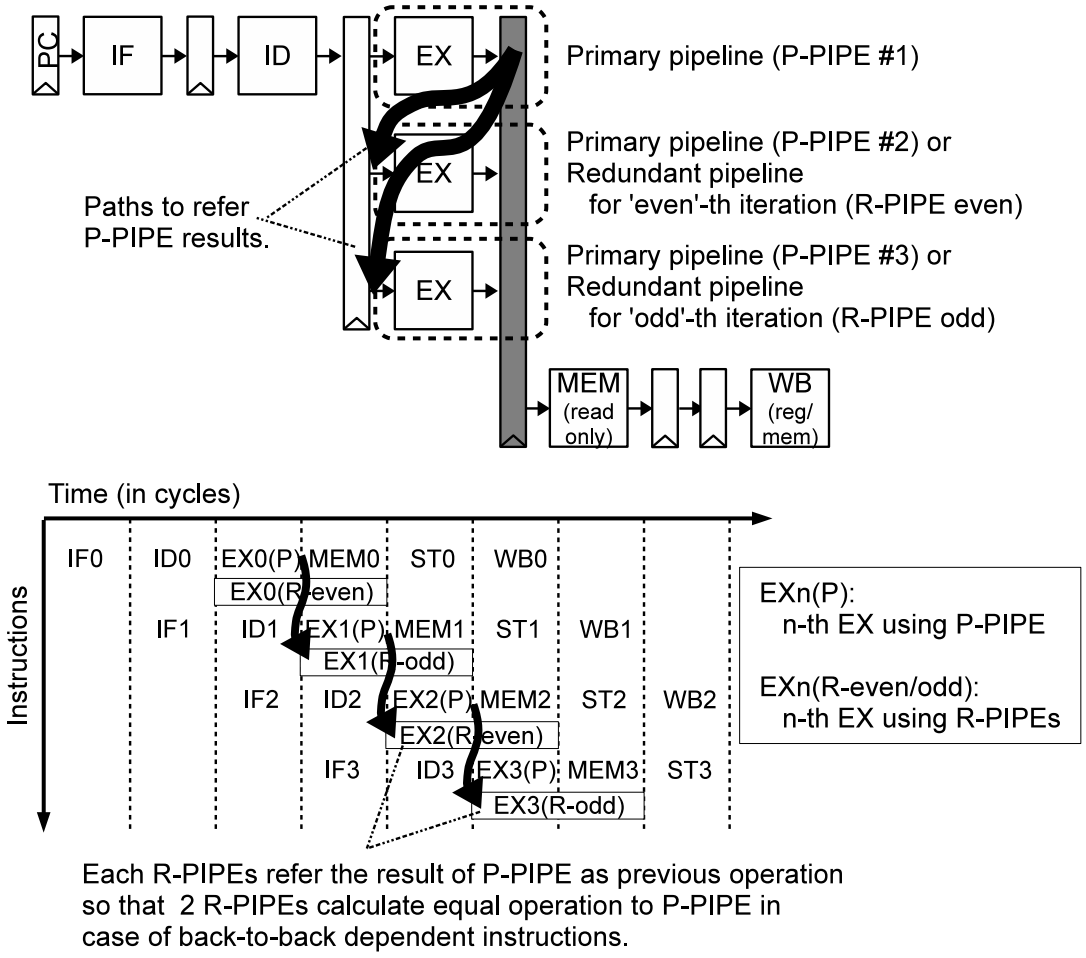


Figure 4.2. P-PIPE and R-PIPEs combination.

Fig. 4.3 shows the pipeline behaviors under a setup error occurrence. We show a comparison between a traditional non-redundant method and the redundant data-path method. Fig. 4.3 (a) is the case of the conventional method in the paper [9]. Assume that the setup error is detected by a Razor-FF in the execution stage 'EX1*'. The succeeding stages will be discarded, as 'flush' in Fig. 4.3 (a), before the re-execution of instruction 1 is started several cycles later. The penalty for recovery is same as a branch miss-prediction. Under a frequent setup error occurrence around the balancing point, the processor may keep retrying the same instructions with a near zero throughput.

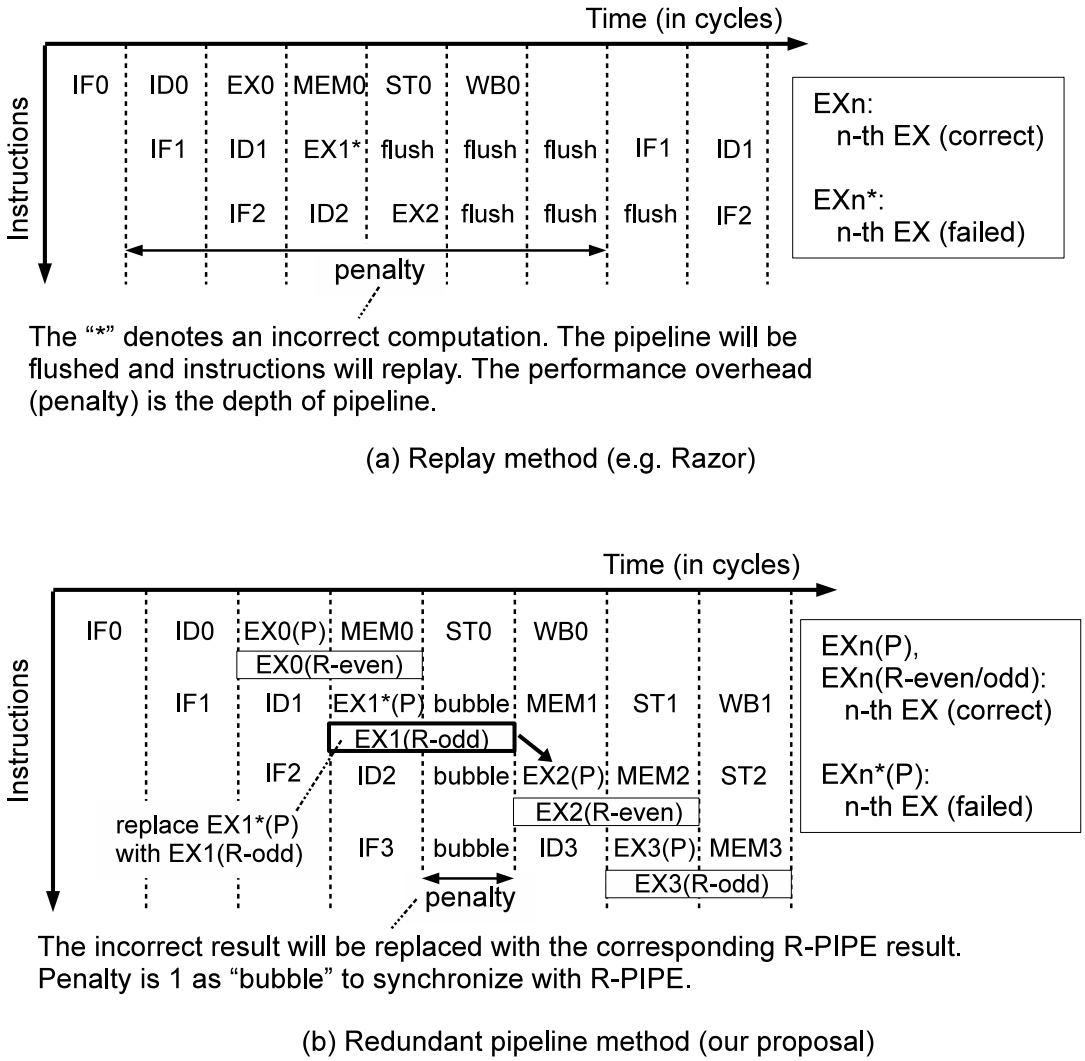


Figure 4.3. Penalty cycles in pipeline (conventional, proposal).

Alternatively, Fig. 4.3 (b) shows the case of recovery in the proposed method. After the timing error is detected in 'EX1*', although the output of the P-PIPE cannot be reliable, we can still use the EX1 (R-odd) output from R-PIPE after one cycle as the correct result. This is because the EX1 (R-odd) is given a period of two cycles to finish the calculation, which can thus be regarded as an error-free timing calculation. In this case, the EX2 in the P-PIPE will use the EX1 (R-odd) output by properly setting the bypassing route. R-PIPE (R-even) also uses the same output by the bypassing route as well. A one-cycle hazard, as 'bubble' in Fig. 4.3 (b) will be inserted to help take the correct value. Compared to the traditional Razor-FF usage, it is supposed that the redundant data-path will maintain some throughput even under a large timing error rate.

1.2 Adapting Redundancy

Apart from the Redundant mode, the three pipelines can use the same working frequency to process three different instructions at most per cycle, which represents a traditional Parallel mode. Because the Redundant mode trades computation resources for better timing error coverage, it must be used under a relatively large timing error rate to achieve good energy efficiency.

Fig. 4.4 shows a program example and the method for switching between Parallel and Redundant execution modes. The program is shown in Fig. 4.4 (a). We use a VLIW-like format to express the idea of either parallel or redundant execution. Suppose that OP6 and OP7 are operations that have longer critical paths, such as multiplication. They will be preset to use Redundant mode. Other operations, from OP1 to OP5, can employ a full instruction level parallelism that is the same as a three-issue processor.

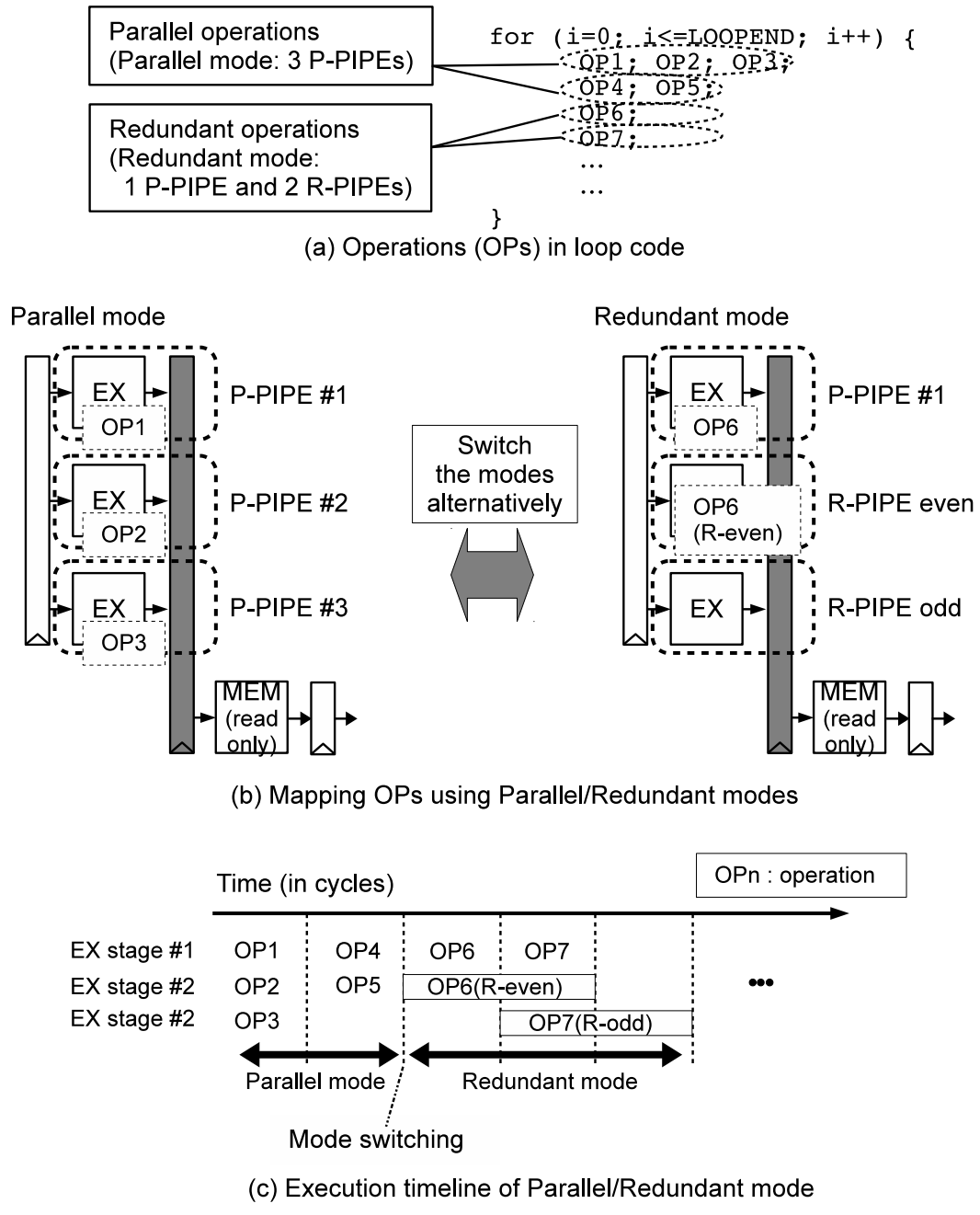


Figure 4.4. Adaptive redundancy using Parallel/Redundant mode switching.

Fig. 4.4 (b) shows the corresponding hardware unit occupation of the above instruction block. OP1, OP2, and $\cdots$, will be issued onto the three pipelines in parallel. After issuing OP5, there will be a very fast mode switch to the Redundant mode. OP6 will be set on both P-PIPE and R-PIPE (even). OP7 will then be assigned to P-PIPE and R-PIPE (odd) in the next cycle. Note that the mode switch will not require a clock change. The only difference between Parallel and Redundant modes is that, in R-PIPE, the clock enables signals to the pipeline registers. As these signals are easy to change by correct selections, no additional hazard is assumed for mode switch in this research. Accordingly, Fig. 4.4 (c) gives the time-line of the instruction execution in the three pipelines.

2. Scheme of determining redundancy

As introduced in Section 1, the Redundant mode will only be required for operations of relatively long critical paths. The Architectural Vulnerability Factor (AVF) is a well-known metric to study the influence of different program behavior patterns on soft error, as introduced in papers [17, 18, 19, 20]. However, AVF does not have factors related to the delay of function units. It is relatively hard to directly measure the vulnerability of setup timing error. For this purpose, we propose a Delay Criticality Factor (DCF) to measure risks related to the delay of operating units. The definition of DCF and risks of malfunction are described as follows:

$$\mathrm{Pr}_{\mathrm{path}} = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{t_{\mathrm{delay}} - t_{\mathrm{typdelay}}}{\sqrt{2\sigma^2}}\right)\right) \tag{4.1}$$

$$\mathrm{DCF} = \frac{\sum\left(\mathrm{Pr}_{\mathrm{path}} \times \mathrm{N}_{\mathrm{path}}\right)}{\#B} \tag{4.2}$$

$$\mathrm{RISK}_{\mathrm{setuperr}} = \mathrm{ERR}_{\mathrm{setup}} \times \mathrm{DCF} \tag{4.3}$$

In the above equations, $\mathrm{Pr}_{\mathrm{path}}$ is the probability of setup errors caused due to the delay of each critical path t_{delay}. In each operation unit (e.g. ALU in data-path), the $\mathrm{Pr}_{\mathrm{path}}$ of the activated path can be used to indicate its current setup error possibility. Equation 4.1 assumes that the delays from all critical paths follow a normal distribution. According to Eq. 4.1, $\mathrm{Pr}_{\mathrm{path}}$ will be 100% in the case that

t_{delay} is the same as the maximum delay in all critical paths. In Eq. 4.2, N_{path} denotes the amount of nets included in each critical path. The N_{path} serves as the weight of the corresponding path. $\#B$ is the amount of all nets included in the operating unit. The DCF is thus a weighted average gotten by combining the Pr_{path} of each path. Each operation unit will have a corresponding DCF that measures its vulnerability to setup error due to the delay distributions of its circuit. In Eq. 4.3, $\text{ERR}_{\text{setup}}$ is the setup error rate observed in the system (e.g. by error detection flip-flops). The risk of malfunction of each operation is denoted as $\text{RISK}_{\text{setuperr}}$, which is the product of the corresponding DCF of that operation unit and the $\text{ERR}_{\text{setup}}$.

As DCF determines the vulnerability of each operation to the setup error, it is possible to use this value to switch adaptively between Parallel and Redundant modes due to the detailed instruction flow. A DCF threshold is used to define the boundary in order to switch the mode, as in Eq. 4.4. The DCF of each operation will be obtained by the circuit level delay analysis. Combining a predefined risk level $\text{RISK}_{\text{threshold}}$ and the current error $\text{ERR}_{\text{setup}}$ given by a dynamic detection from Razor-FF, the redundancy-adapting scheme is shown in Fig. 4.5.

$$\text{DCF}_{\text{threshold}} = \frac{\text{RISK}_{\text{threshold}}}{\text{ERR}_{\text{setup}}} \tag{4.4}$$

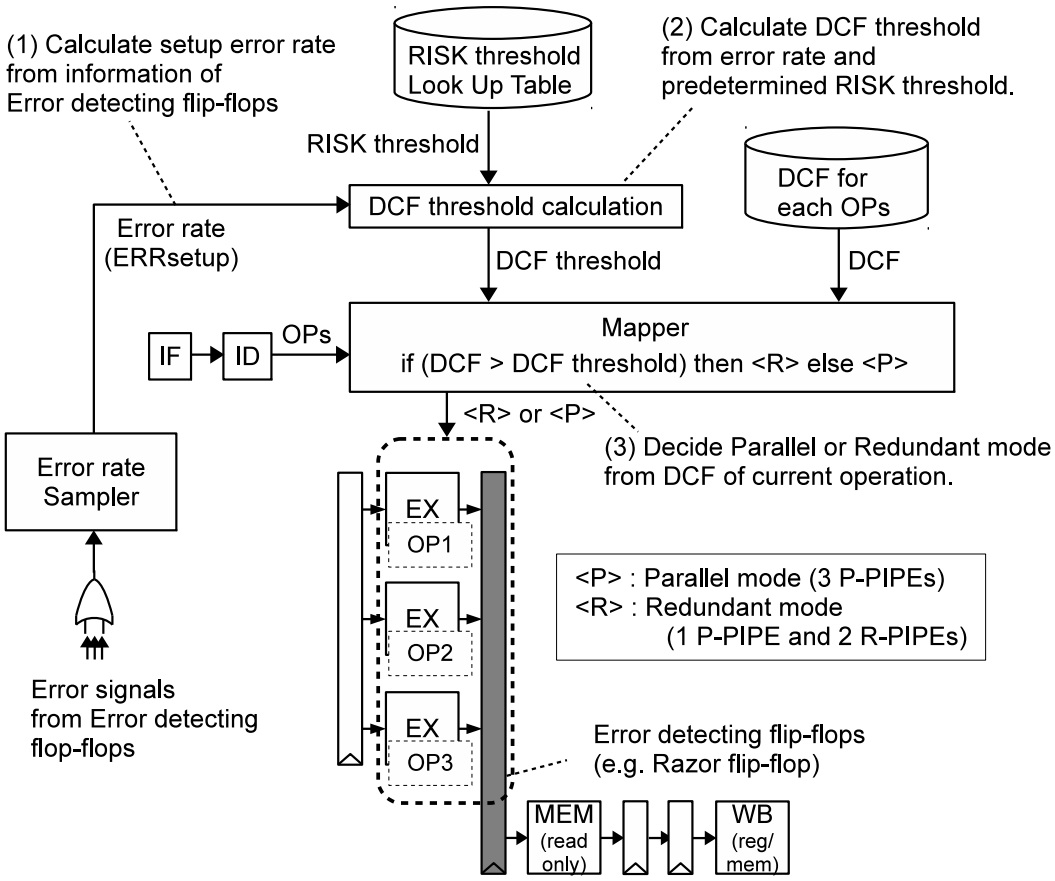


Figure 4.5. Determine redundancy by DCF, $\mathrm{ERR}_{\mathrm{setup}}$ and $\mathrm{RISK}_{\mathrm{threshold}}$.

As in Fig. 4.5, the error rate sampler gathers error detection signals generated from error detecting flop-flops. The value of $\mathrm{ERR}_{\mathrm{setup}}$ is then calculated according to the collected errors, following (1) in Fig. 4.5. The risk of malfunction $\mathrm{RISK}_{\mathrm{threshold}}$ is predetermined and stored in a look-up table. Equation 4.4 is used to get $\mathrm{DCF}_{\mathrm{threshold}}$ in the DCF threshold calculation unit, as in (2) in Fig. 4.5.

Assume that we are using a reconfigurable processor that maps instructions to the execution units prior to the actual execution. During the mapping, the mapping unit (Mapper) looks at the OP, its DCF and $\mathrm{DCF}_{\mathrm{threshold}}$. DCFs are also provided by a look-up table with an index corresponding to operation type. The Mapper then determines the need of a Redundancy mode. Note that in a non-reconfigurable processor, the above determination can be performed at the decode phase.

Several additional hardware components will be necessary to implement RazorProtector. Fig. 4.6 summarizes area overhead in this method. As shown in Fig. 4.3, under the Redundant Mode, control logics are required to forward correct execution results in the R-PIPE to the erroneous P-PIPE, triggered by the error detection in the Razor-FF of P-PIPE. Also, as shown in Fig. 4.5, new logic will be added into the instruction decode phase, to give a proper determination of the adaptive redundancy. This logic starts from the calculation of the DCF according to the instruction type, and generates a control signal to select either the Parallel or the Redundant Mode by studying the possibility of a setup error under the current risk level. Accordingly, several counters, comparators and LUTs are used for this redundancy control logic. The evaluation results indicate that these additional logics will introduce a 1.4–1.7% increase and cause 0.5% additional power for logics. LUTs can minimize the storage size of the DCF table, which will be updated dynamically like a cache memory. The experiments show that a set of instruction types, with a number of less than 100, appears in normal functions. A table of 256 entries is thus sufficient for the DCF lookup purpose, indexed by the instruction type. This condition allows the table to be updated only at the beginning of the program. The additional memories will introduce a 1.0% increase, which causes 1.1% additional power for memories.

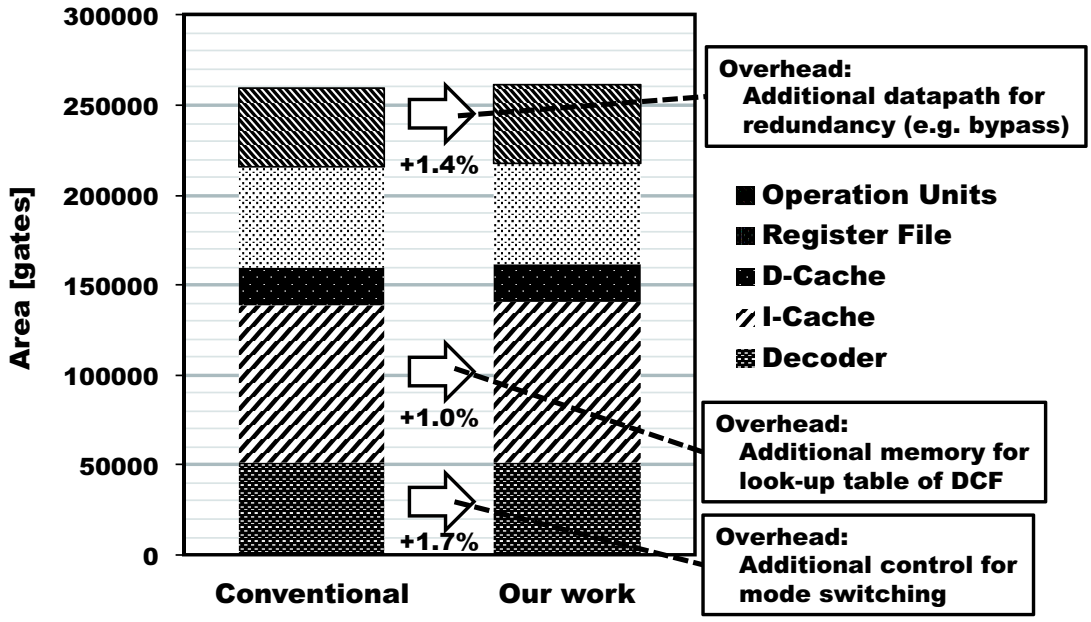


Figure 4.6. Area overhead for each logic/memory part.

3. Evaluation Methodology

This section introduces the mathematical calculation based evaluation in which we theoretically present a processor performance model and its probability based response to redundancy rate under different setup error rate. The mathematical method is then verified by a cycle accurate simulation, which gives a more dynamic and accurate relationship among the performance, program characteristics and error rates. Detailed results of RazorProtector will be introduced in Section 4 and 5, regarding the mathematical model and the cycle accurate simulation.

Fig. 4.7 shows the evaluation framework, which is based on a mathematical model of the denoted pipeline architecture and processor simulator. The corresponding circuit data is extracted from a special FR-V processor [21].

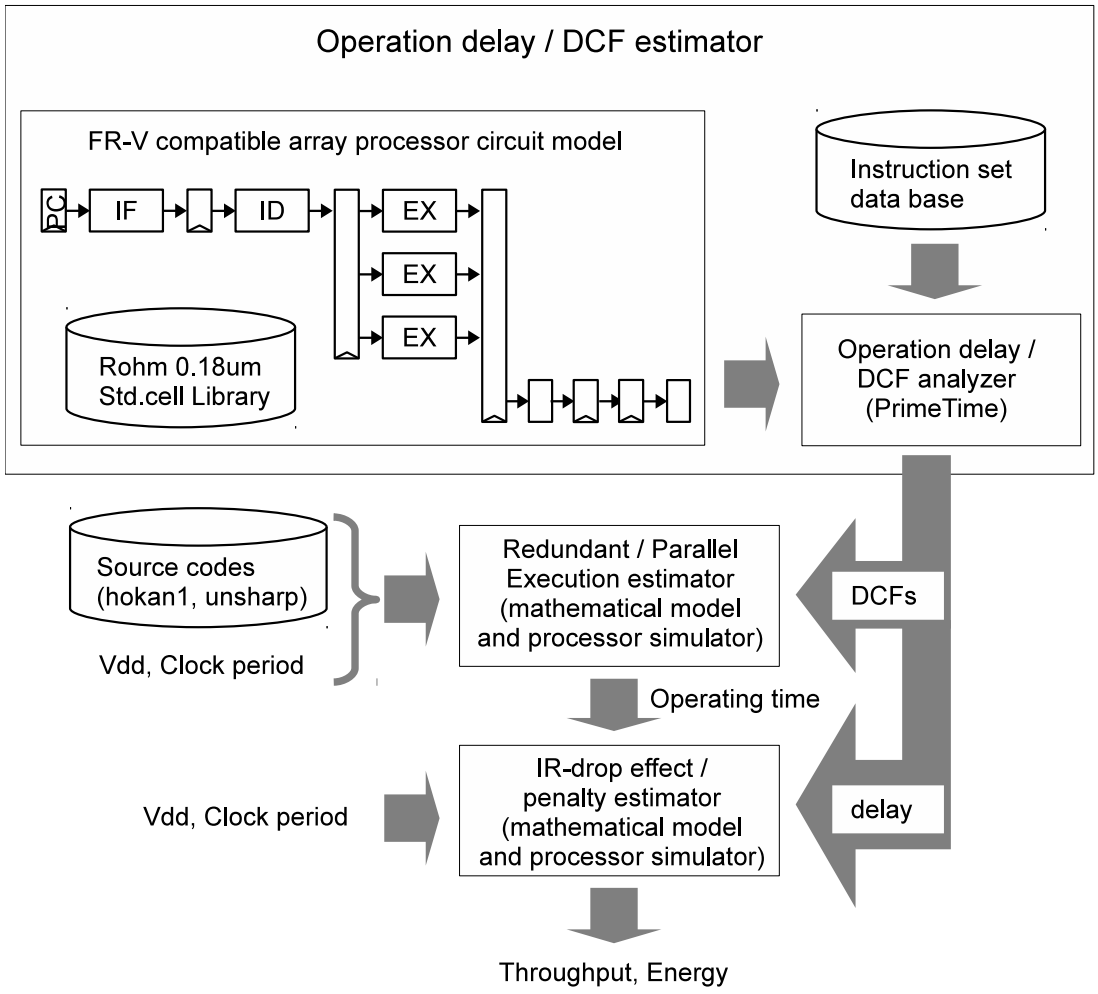


Figure 4.7. Evaluation framework.

This dissertation assumes that each VLIW instruction in the FR-V ISA may contain three arithmetic executions at most. The evaluation framework has following parts:

1. Operation delay/DCF estimator, using a static timing analyzer (Synopsys PrimeTime) to extract operation delay corresponding to the instruction and the operation mode. The delay is obtained by using a Rohm 0.18um standard cell library and a wire load model. DCFs are calculated from delays through the static timing analyzer.

2. Redundant/Parallel Execution estimator, which emulates the determination of Redundant/Parallel modes and calculate operation time based on a mathematical model and a processor simulator.

3. IR-drop/penalty estimator, which emulates the penalty due to IR-drop based on the mathematical model and a trace-based simulation. Throughput and energy consumption can be obtained by using operation time and penalties due to setup errors.

Following equations are used to get performance results.

$$T_{\mathrm{all}}(V_{\mathrm{dd}}) = T_{\mathrm{op}} + T_{\mathrm{rcv}}(V_{\mathrm{dd}}) \tag{4.5}$$

$$T_{\mathrm{op}} = \frac{n}{n(1-R) + \frac{n}{3}R} \cdot t_{\mathrm{op}} \tag{4.6}$$

$$\begin{aligned} T_{\mathrm{rcv}}(V_{\mathrm{dd}}) = {} & k_1 \cdot {}_{(n(1-R))}H_3 \cdot \mathrm{err}(V_{\mathrm{dd}}) \cdot t_{\mathrm{err}} \\ & + k_1 \cdot {}_{(\frac{n}{3}R)}H_3 \cdot \mathrm{err}(V_{\mathrm{dd}}) \cdot t_{\mathrm{err}}(1-B) \end{aligned} \tag{4.7}$$

$$\mathrm{err}(V_{\mathrm{dd}}) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{\mathrm{delay}(V_{\mathrm{dd}}) - t_{\mathrm{typdelay}}}{\sqrt{2\sigma^2}}\right)\right) \tag{4.8}$$

$$\mathrm{delay}(V_{\mathrm{dd}}) = k_2 \cdot \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}} \tag{4.9}$$

(Notes: erf is error function)

Eq. 4.5 gives operation time for n operators at working voltage V_{dd}, in which the 1st term is the error-free execution time, and the 2nd term is the penalty time due to setup errors. These two terms will be introduced respectively in Eq. 4.6 and Eq. 4.7 in detail.

R in Eq. 4.6 is the redundancy ratio, which ranges from 0 to 1. t_{op} is the error-free execution time per each unit operation.

In Eq. 4.7, the 1st term shows the execution time penalty due to the setup error in non-redundant mode. The penalty is produced as a combined effect of redundancy ratio R and the real instruction sequence under each redundancy. When the previous instruction is erroneously executed and it takes one more cycle before obtaining the correct results from the R-PIPE, the following dependent instruction will also be delayed. We assume that data from $n(1-R)$ operations are fed into the three pipelines, following the term of a repeated combination $({}_{(n(1-R))}H_3)$. k_1 is the correction factor needed to reflect the actual effective paths. This dissertation assumes that the maximal issue width is 3 and that half of the results of computation are not referred to in the succeeding operations. Accordingly, n is defined as 3 and k_1 is defined as 0.5.

In Eq. 4.7, $\mathrm{err}(V_{\mathrm{dd}})$ is the error rate at the working voltage V_{dd} which will be introduced in Eq. 4.8. t_{err} is the time penalty from setup errors, which corresponds to the period up until the time that the voltage is re-adapted to a higher level to avoid setup errors. t_{err} depends on the voltage scaling speed and the re-execution operation cycles. B is the penalty reduction ratio from the effect of redundant data-path.

Eq. 4.7 gives the execution time under the redundant operation mode by using both P-PIPE and R-PIPEs. The data-path dependence is also taken into account as well as the non-redundant execution in the 1st term. As all the three pipelines work on the same instructions under Redundant mode, the effective throughput is 1/3 of the Parallel execution, expressed by the repeated combination $({}_{(\frac{n}{3}R)}H_3)$.

Eq. 4.8 shows the error rate in detail. We assume that the unit operation delay takes a normal distribution around the average delay t_{typdelay}. When the delay exceeds the design constraints and the corresponding path is enabled in the execution, the operation will result into a setup error. The cumulative distribution of the whole data-path can be referred as the error rate, as shown in Eq. 4.8.

Eq. 4.9 gives the delay at working voltage V_{dd}, using the alpha-power law delay model [15]. The coefficient α and k_2 are defined as 1.3 and 1 according to the 0.18um process technology we use [16].

The throughput Throughput(V_{dd}) and energy Energy(V_{dd}) at working voltage V_{dd} are expressed as follows.

$$\mathrm{Throughput}(V_{\mathrm{dd}}) = \frac{1}{T_{\mathrm{op}} + T_{\mathrm{rcv}}(V_{\mathrm{dd}})} \tag{4.10}$$

$$\mathrm{Energy}(V_{\mathrm{dd}}) \propto V_{\mathrm{dd}}^2 \cdot (T_{\mathrm{op}} + T_{\mathrm{rcv}}(V_{\mathrm{dd}})) \tag{4.11}$$

The voltage scaling takes the time in the order of microseconds in modern processors, which is 1000 cycles when working frequency equals 100MHz. Accordingly, t_{op} is defined as 1, and t_{err} is defined as 1000 at maximum. In evaluation, t_{err} is linearly scaled corresponding to the voltage. IR-drop is reflected as voltage in this model.

4. Applicability of Adaptive Redundancy

This section discusses the effectiveness of the adaptive redundancy in RazorProtector. A mathematical model is used to give an understanding of conditions where RazorProtector best applies during IR-drops.

4.1 Determining redundancy from DCF

We divide setup error rates $\mathrm{ERR}_{\mathrm{setup}}$ into 7 groups (<4%, <6%, $\cdots$). Fig. 4.8 shows the corresponding $\mathrm{DCF}_{\mathrm{threshold}}$, under the assumption that $\mathrm{RISK}_{\mathrm{threshold}}$ is always 4.0%. $\mathrm{RISK}_{\mathrm{threshold}}$ is actually affected by the dynamic behavior (e.g. frequency and voltage control). This point is also discussed in Section 5.

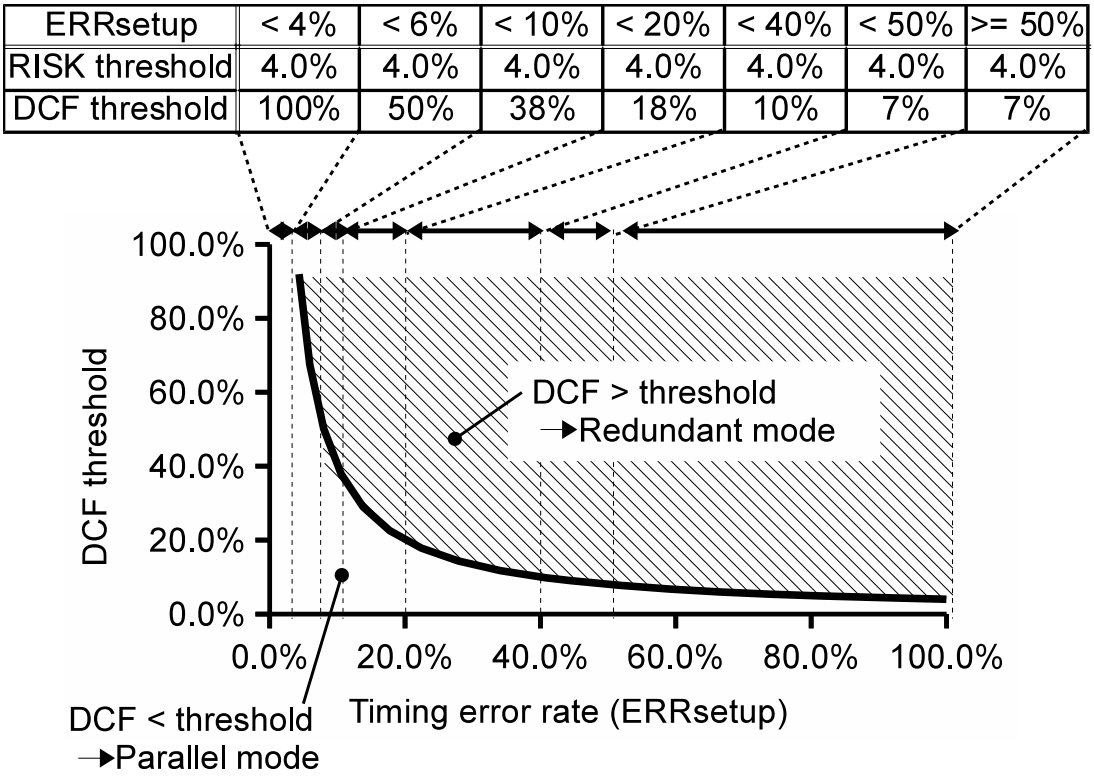


Figure 4.8. Relationship among DCF, $\mathrm{ERR}_{\mathrm{setup}}$ and $\mathrm{RISK}_{\mathrm{threshold}}$.

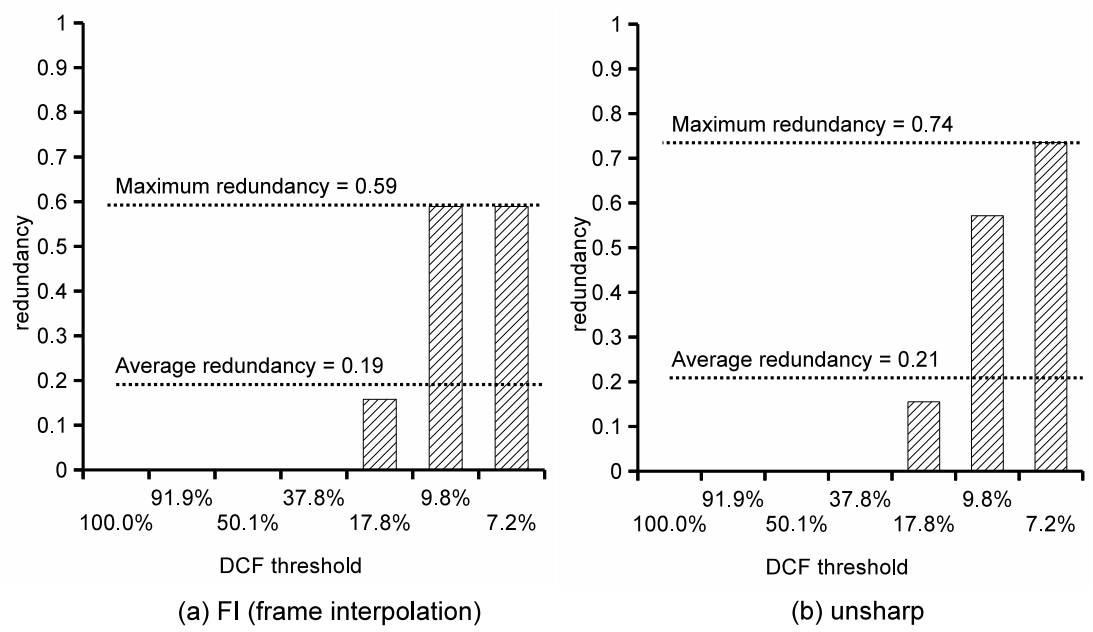


Figure 4.9. Redundancies for different DCF thresholds.

This research uses two filtering functions *FI* and *unsharp* from image processing programs as the workloads to study the effectiveness of the proposed method. Function *FI* is the SAD (Sum of Absolute Difference) part of a frame interpolation, in which most instructions are addition, subtraction, and multiplication. Function *unsharp* performs masking processing including both linear and nonlinear filtering. The ratios of multiplications and shifts in *unsharp* are larger than *FI*. The distribution of operations with different delays can be regarded as a program behavior pattern, since this distribution actually determines the vulnerability to the setup error. Fig. 4.9 gives the redundancy results of these two benchmarks under different DCF thresholds, varying from 100.0% to 7.2%. The following observations can be obtained from Fig. 4.8 and Fig. 4.9. These results are based on the mathematical and trace-based simulation.

1. The Redundancy mode is required only under low DCF thresholds, which represent relatively large $\mathrm{ERR}_{\mathrm{setup}}$ zones as shown in Fig. 4.8.

2. The required redundancy of *unsharp* reaches 0.74 under a 7.2% DCF threshold, which is 25% larger than the maximum redundancy of *FI*. This is because *unsharp* contains more long-delay operations such as multiplications than *FI*. At large $\mathrm{ERR}_{\mathrm{setup}}$ zones, these long delay instructions will bring a dominant contribution to the setup error.

3. However, *unsharp* also contains more short-delay operations such as shifts than *FI* does. For this reason, the two functions have similar redundancy requirements, on average. It can also be expected that *unsharp* is more sensitive to the DCF threshold than *FI* is.

4.2 Effectiveness of adaptive redundancy

This section shows the first study the effectiveness of a redundant data-path based on a fixed redundancy. Fig. 4.10 and 4.11 show the results of energy consumption and performance by comparing the proposed method to the traditional DVS method, which is based on the mathematical and trace-based simulation. The x-axes in these figures are the supply voltage values, normalized by V_{opt}. We assume that an unexpected IR-drop will make the supply voltage go from

1.0 toward 0.9. The corresponding setup error rate during this period is calculated by following the alpha-power law delay model [15, 16]. The y-axes show the normalized energy consumption and performance in Fig. 4.10 and Fig. 4.11, respectively. Two sets of experiments are derived from Eq. 4.5 - Eq. 4.9. These are conducted by applying different redundancy ratio R as fixed high and low redundancies, denoted respectively as the (a) and (b) parts of each figure.

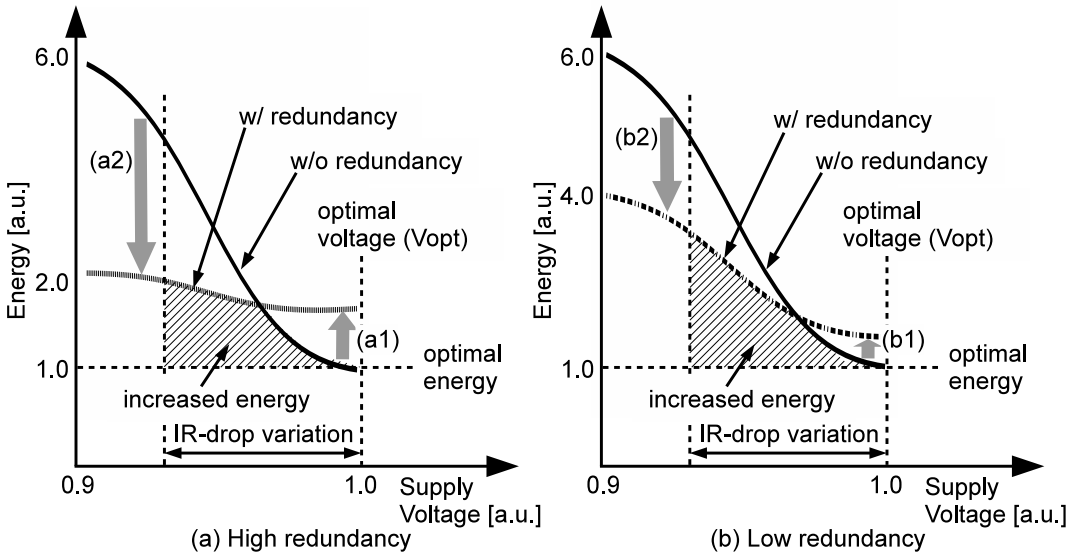


Figure 4.10. Increase of energy consumption due to IR-drop.

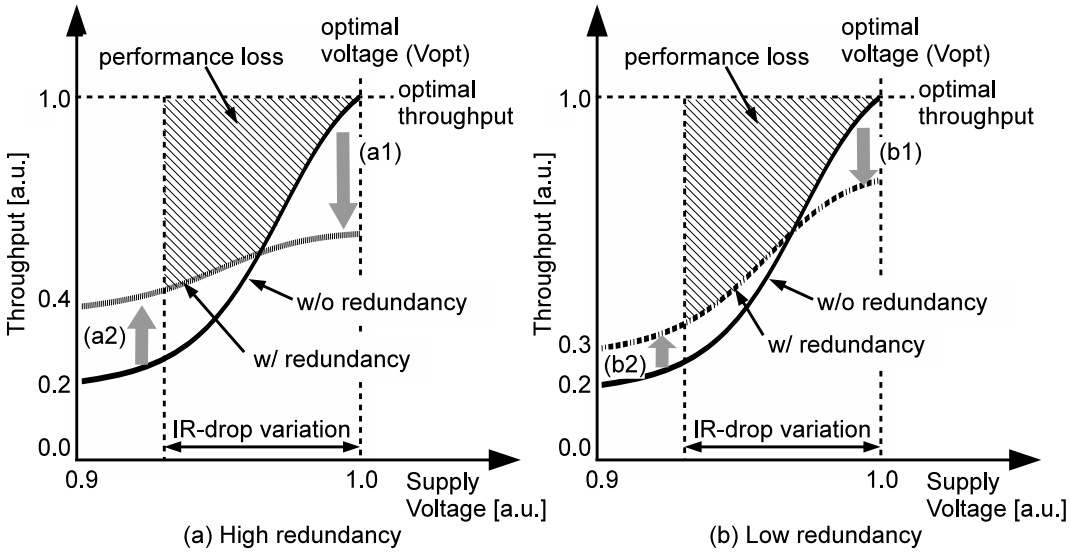


Figure 4.11. Performance loss due to IR-drop.

It can be observed from Fig. 4.10 that the redundant data-path will have a higher energy consumption value than the traditional method at regions near V_{opt}, as indicated by the arrow (a1). The redundancy performs unnecessary calculations when the original data path can give sufficiently correct results under a low error rate. The benefits from the redundant calculation outweigh the cost when the error rate increases to a certain level, indicated by the (a2) trend in Fig. 4.10. The shaded regions in Fig. 4.10 actually indicate the zones where a redundant data-path should be applied to get better energy consumption than with the traditional method. As explained in Chapter 3, it will be in the order of microseconds in modern processors, many cycles will be lost before the new balancing voltage is reached. It is the major reason that the redundant calculation is effective rather than the parallel calculation under a high error rate.

Fig. 4.10 (b) shows a trend similar to Fig. 4.10 (a) towards reducing energy consumption under a large error rate. However, due to a lower redundancy than Fig. 4.10 (a), the energy increase (b1) at a low error rate and the energy saving (b2) at a high error rate are both smaller than in Fig. 4.10 (a). It is thus necessary to adapt redundancy to obtain the best energy consumption. Fig. 4.11 has a same format as Fig. 4.10. Similar conclusions can be drawn for the performance study of the proposed DVS method.

The results of the DVS control based on an adaptive redundancy are given in Fig. 4.12. The horizontal axes in these two figures show the statistical variation of the unexpected IR-drop, which varies from 1% to 10% of the optimal voltage V_{opt}. The setup error rate caused by the IR-drop will increase along the horizontal axis. The solid and dashed lines in Fig. 4.12 respectively depict the increases of energy consumption due to setup errors in the traditional method and in the adaptive redundancy. The redundancy of the proposed method is adjusted by following the scheme in Section 2.

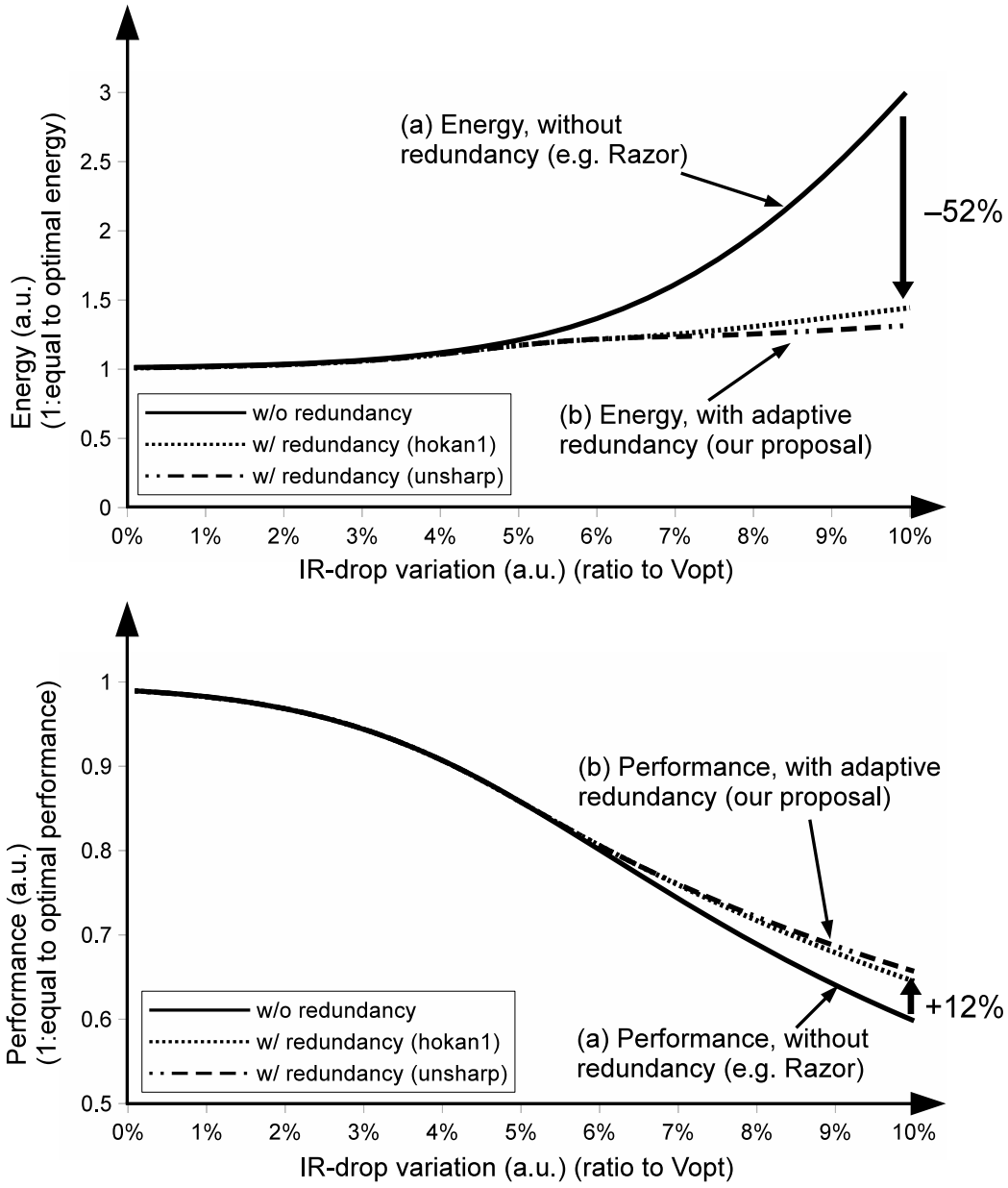


Figure 4.12. Results of the adaptive redundancy.

As shown in Fig. 4.12, without the support from a redundant execution, a traditional DVS method with Razor-FFs cannot tolerate an unexpected IR-drop. The increase of energy consumption will reach 200%, as a penalty, when IR-drop becomes 10% of the balanced supply voltage. However, the redundancy controlled by a DCF study of the instruction scheme can be raised effectively in large error zones. As in Fig. 4.12, the energy consumption is 52% less than the value in the non-redundant method. Also, different from Fig. 4.10, the DCF based method can turn off redundancy under a low error rate. As in Fig. 4.12, the solid and dashed lines are relatively flat and demonstrate little difference in small IR-drop zones, which indicates that a near zero redundancy is set by the control scheme.

The performance results seen in Fig. 4.12 show the same tendency as the energy curve. In detail, a 12% performance loss can be avoided with the proposed adaptive redundancy method under a frequency setup error occurrence. The results of functions *hokan1* and *unsharp* are very similar in Fig. 4.12. This can be explained by the maximum redundancy part of Fig. 4.9. Function *unsharp* will require a higher redundancy than *hokan1* in high error rate zones. Combining this with energy saving curves in Fig. 4.10, the execution of *unsharp* that follows (a2) will show a relatively greater effectiveness than the non-redundancy mode.

With a lower energy consumption increase and a lower performance loss, the proposed redundancy adapting method can help further reduce the margin in traditional DVS methods. For example, a 10% voltage margin may be necessary in traditional methods to avoid a sudden energy increase when an unexpected IR-drop occurs. The RazorProtector method can remove 10% margin by adapting redundancy, which in return achieves 20% smaller energy consumption.

5. Practically Simulated EDP results

The previous section shows physically that RazorProtector with an adaptive redundancy works to reduce energy and performance loss in large IR-drop zones. In this section, based on a cycle accurate simulator, we give an estimation of RazorProtector from the Energy-Delay-Product (EDP) measure, assuming that the preferred platform is a workstation or mobile devices and so on, where EDP applies best. The simulation platform is described in detail in Tab. 4.1. Param-

eters in the RazorProtector are tuned, in this simulation environment, to get an optimal EDP reduction. More influences are added into the simulation to reflect a practical environment.

Table 4.1. Parameters of cycle-based simulator.

Instruction issue width	4/cycle
No. of General purpose register (GR)	32
No. of Media register (MR)	32
L2 ⇒ L1 transferring	8bytes/cycle
Instruction cache	4way 16KB (64bytes/line)
L1 cache	4way 16KB (64bytes/line)
L1 ⇒ L0 transferring	12bytes/cycle
Store buffer	4entry

5.1 Tuning risk threshold

After the IR-drop happens, the timing error rate will remain at a relatively high value before the voltage is re-adapted to the balanced level. Therefore, the voltage scaling up speed directly affects the working model of RazorProtector. In some extreme cases, if the voltage scaling up penalty is 0, the traditional Razor-FF processor will have the best EDP, since the re-adapting can be finished in the next cycle. If voltage scaling is extremely slow, the redundant mode should always be applied by setting a near 0 allowed risk level, which is $\text{RISK}_{\text{threshold}}$. In this section, we give results of $\text{RISK}_{\text{threshold}}$ value by exploring several possible voltage scaling speeds.

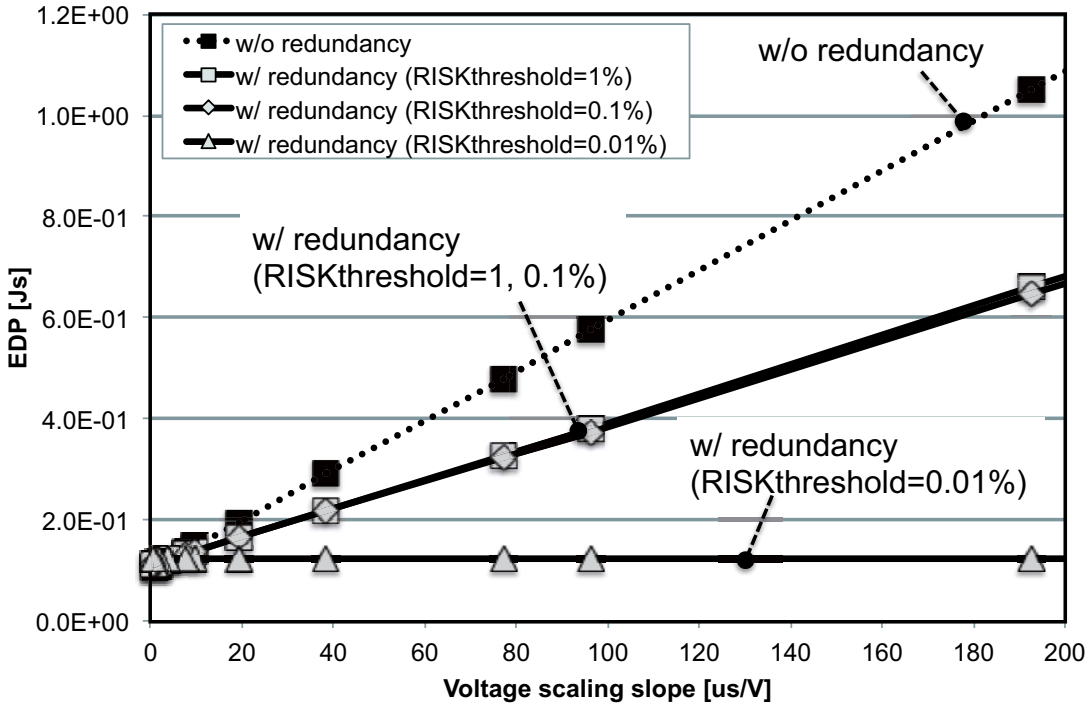


Figure 4.13. EDP results of different $\text{RISK}_{\text{threshold}}$ (FI, 10% IR-drop).

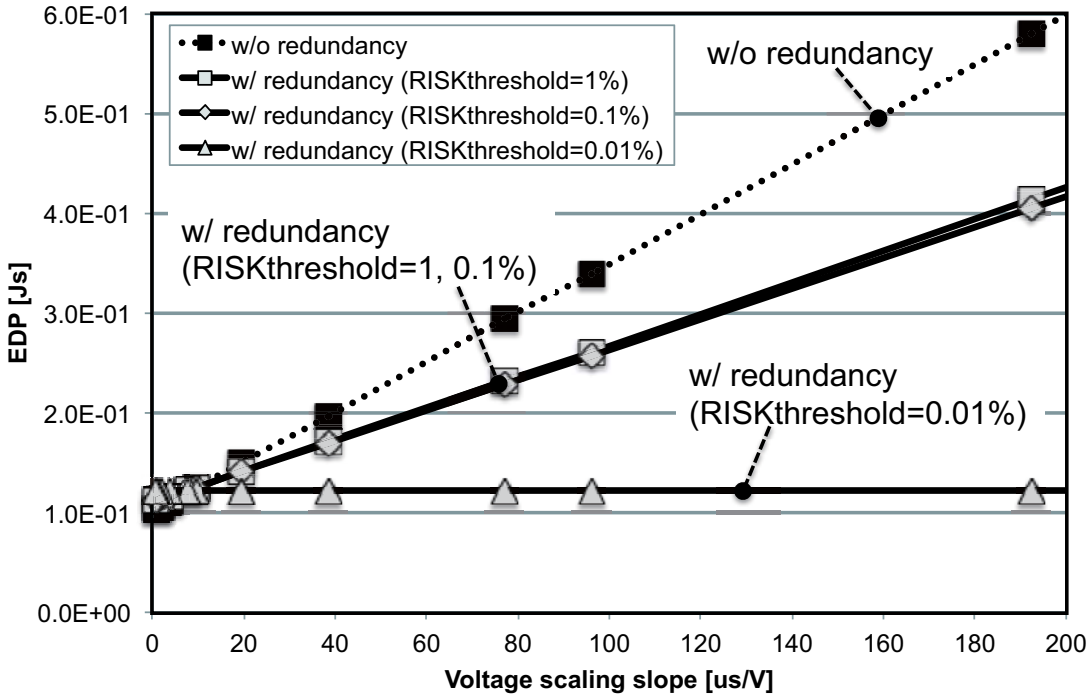


Figure 4.14. EDP results of different $\mathrm{RISK}_{\mathrm{threshold}}$ (FI, 5% IR-drop).

We have conducted a series of experiments, studying the EDP results under several $\text{RISK}_{\text{threshold}}$ values. Three $\text{RISK}_{\text{threshold}}$ values 0.01%, 0.1% and 1% are tried under high and low IR-drop conditions that are 10% and 5%, respectively. The simulation environment generates IR-drop stimulus randomly, where 10% and 5% conditions are maximum IR-drop in experiments. The EDP results of the corresponding execution of functions *FI* are shown in Fig. 4.13 and Fig. 4.14, where Fig. 4.13 uses the 10% and Fig. 4.14 takes the 5% IR-drop. The vertical axes in the two figures are EDP results. The voltage scaling speed is given in the horizontal axes, which is the time duration used in each unit voltage scaling. It affects the time cost that a new voltage balancing can reach. The left most value 0μs/V is the most ideal voltage scaling situation, while the rightmost value represents a relatively practical scaling in current processors. It is assumed that the redundancy mode is used only before the voltage balancing point is re-adapted. In these two figures, we give the results of the RazorProtector, which uses adaptive redundancy, and conventional Razor based DVS processors without R-PIPEs as a comparison.

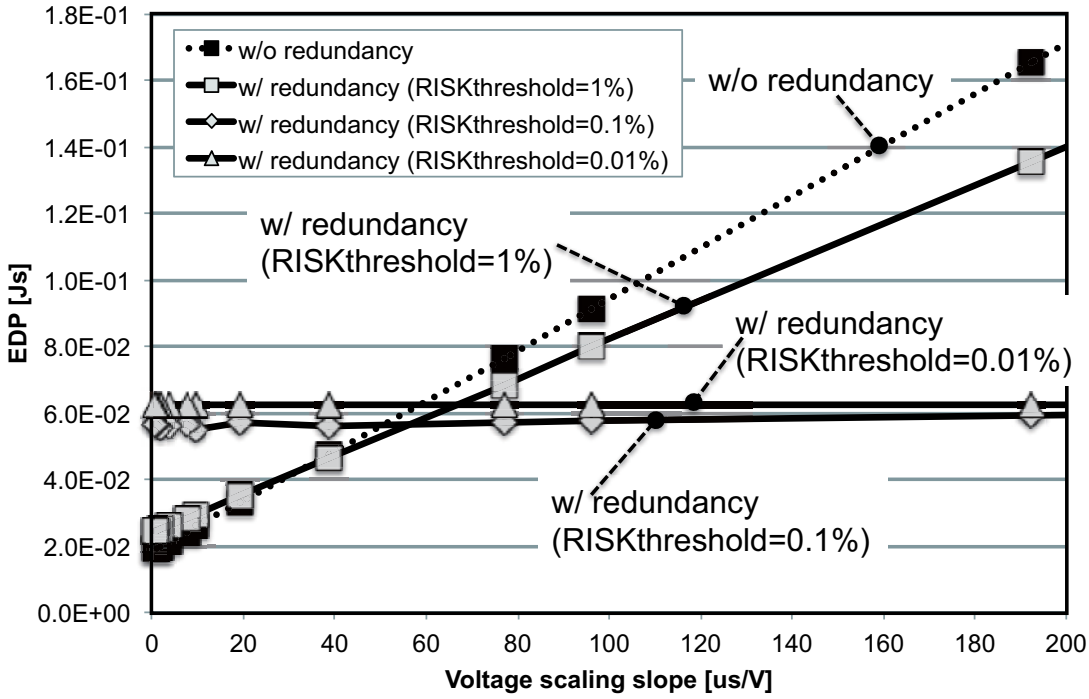


Figure 4.15. EDP results of different $\mathrm{RISK}_{\mathrm{threshold}}$ (unsharp, 10% IR-drop).

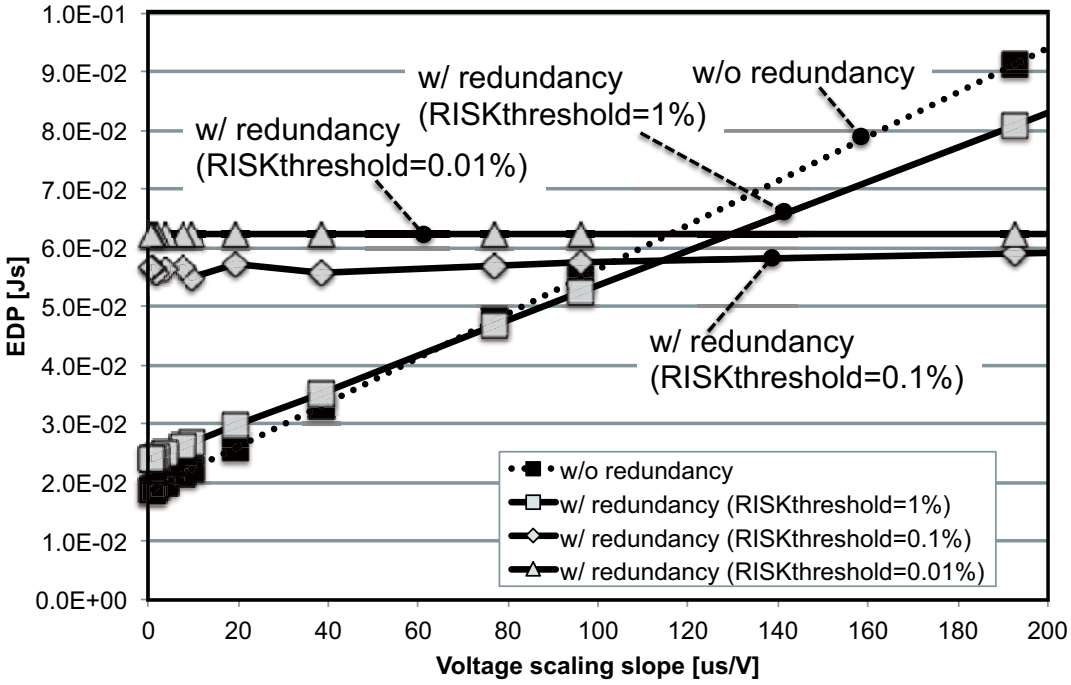


Figure 4.16. EDP results of different $\mathrm{RISK}_{\mathrm{threshold}}$ (unsharp, 5% IR-drop).

It can be observed from Fig. 4.13 and Fig. 4.14 that under any conditions, RazorProtector performs better than the original Razor DVS application. Considering the three $\text{RISK}_{\text{threshold}}$ values, in both IR-drop rates, the change of voltage scaling has no effect on the EDP results when the $\text{RISK}_{\text{threshold}}$ equals to 0.01%. It can be imagined that under this threshold, the program execution has been properly duplicated and the redundant data-path provides a good covering of the recovery. Accordingly, even though the processor stays under an insufficient voltage longer when the voltage scaling speed is low, the performance will not be damaged, because the flush-based recovery has been avoided by the redundant data-path. On the other hand, the $\text{RISK}_{\text{threshold}}$ 0.1% and 1% are not quite sufficient for function *FI*. Since the allowed high thresholds generate low redundancy, the flush based recovery in the parallel mode will influence the performance so that both of these two thresholds have larger EDPs than 0.01%. The results of low voltage scaling speed emphasize this observation, as shown in the right zones of Fig. 4.13 and Fig. 4.14[1] .

The situation of function *unsharp* is more complicated than that of function *FI*. Fig. 4.15 and Fig. 4.16 are the EDP results of *unsharp*, which have the same formats as Fig. 4.13 and Fig. 4.14. Basically, $\text{RISK}_{\text{threshold}}$ 0.01% is similar to 0.1% and is better than threshold 1%, which may miss some parts of possible application of the redundancy mode in the low voltage scaling speed zones. However, the results become different in the high voltage scaling speed zones. Even the 100% parallel mode without redundancy outperforms these thresholds when the voltage scaling can be done quickly, especially if the 5% error rate is used. This is because instructions in *FI* concentrate on the stable and similar DCF values while *unsharp* takes an opposite DCF tendency, as has been discussed in Fig. 4.9. Fig. 4.9 shows that *unsharp* has more longer delay instructions than *FI*. But it also contains more short delay instructions. As a result, a very small allowed risk level, represented by $\text{RISK}_{\text{threshold}}$ 0.01%, introduces unnecessarily the redundant mode. A further study indicates that function *FI* has an average IPC of 1.7 under parallel mode while *unsharp* has an average IPC of 2.1. This indi-

[1] Note that the $\text{RISK}_{\text{threshold}}$ in these figures have different values than the mathematical model. An accurate simulation puts the processor under an insufficient voltage frequency pair during IR-drops, which consequently requires a smaller threshold value.

cates that in the processor model, *unsharp* uses execution resources more than *FI*. Therefore, a high redundancy in *unsharp* will cause insufficiency in the processor execution units, making the performance difference between the parallel mode and redundant mode larger in *unsharp* than in *FI*. For all these reasons, $\text{RISK}_{\text{threshold}}$ values of complicated programs should be carefully chosen according to the voltage scaling speed parameter, in order to get an optimal EDP reduction.

5.2 EDP reduction results

Figures 4.13 to Fig. 4.16 have already shown that a good $\text{RISK}_{\text{threshold}}$ can help RazorProtector to determine a proper redundancy during the execution and thus help to maintain a stable EDP even under a 10% or 5% IR-drop. In this section, we give a summary of the EDP reduction that RazorProtector can achieve, as shown in Fig. 4.17. $\text{RISK}_{\text{threshold}}$ 0.01% and 0.1%, respectively, have been used for function *FI* and *unsharp*, regarding the tuning results in Section 5.1. The IR-drop is assumed to be 10% and 5% respectively. The EDP data in this figure have been normalized by the EDPs of 100% parallel mode, which represents a normal Razor-FF processor.

Figure 4.17 shows that RazorProtector can achieve more EDP reduction than the non-redundant mode under a large error rate and a long voltage up-scaling period. For function *FI*, RazorProtector can reduce 37% EDP when the voltage scaling slope is 20μs/V. At a 100μs/V and 200μs/V slope, the EDP reduction result reaches 78% and 88% respectively, under the 10% IR-drop value. This indicates that RazorProtector can achieve a good EDP reduction for a processor with a microsecond order voltage scaling restriction.

The results of function *unsharp* have shown a similar trend. However, in a processor with fast speed voltage scaling, a non-redundant model works better even when the $\text{RISK}_{\text{threshold}}$ is carefully tuned. As described in previous sections, *unsharp* has a high IPC. Under the parallel mode, it can use full resources to boost performance. Accordingly, the performance loss due to instruction duplication becomes more critical if the voltage scaling is not slow. After a 40μs/V voltage scaling slope, EDP reduction is possible in RazorProtector. Under a practical voltage scaling speed like 100μs/V and 200μs/V, 12% and 18% EDP reduction can be achieved by using adaptive redundancy if 10% IR-drop happens.

For both benchmarks, the 5% IR-drop curve shows a tendency very similar to the 10% IR-drop curve. The EDP reduction results become smaller. At the 200μs/V voltage scaling slope, RazorProtector can get 70% EDP reduction for function *FI* and 11% EDP reduction for *unsharp*.

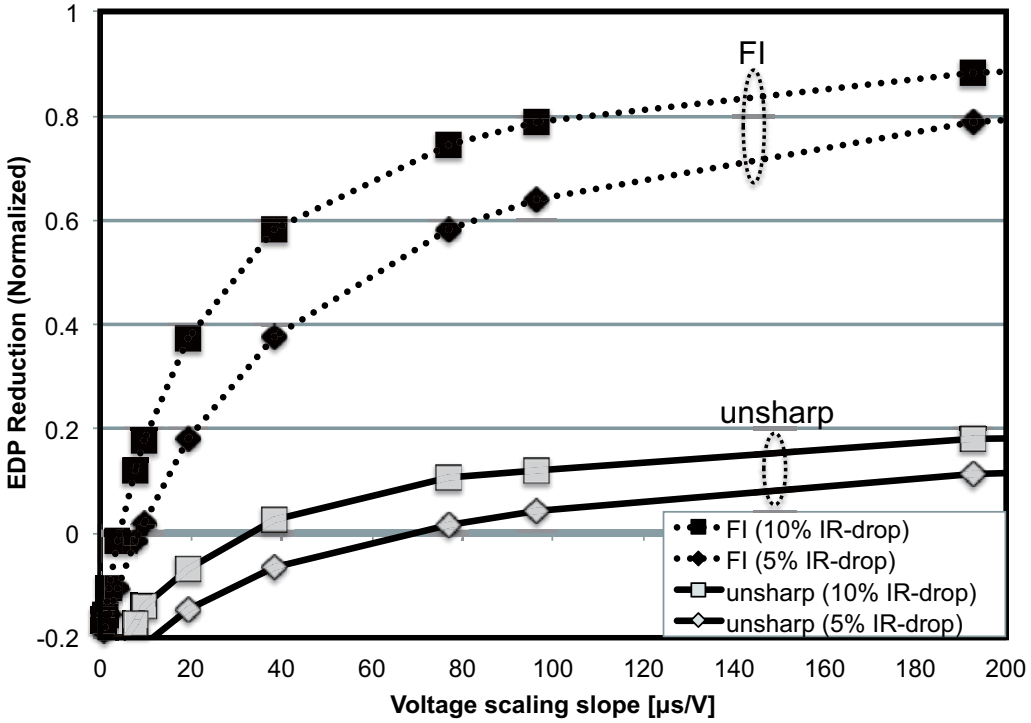


Figure 4.17. EDP reduction by RazorProtector.

6. Conclusions of this Study

This chapter proposed RazorProtector, a redundant data-path based method to help reduce the recovery penalty for processors in which DVS is aggressively applied. Using a special metric DCF to measure the setup error vulnerability, an adaptive redundancy was used to best tolerate unexpected IR-drops at post voltage balancing regions. The results show that under a large setup rate caused by a 10% unwanted voltage drop, the adaptive redundancy can reduce EDP up to 78% at the 100μs/V voltage scaling slope and 88% at the 200μs/V voltage scaling slope. RazorProtector can help maintain Razor energy efficiency for a processor with a microsecond order voltage scaling restriction.

The further work lies mainly in studying the dynamic application of the adaptive redundant method to achieve better toleration of setup error variations. In detail, the relationship between DCF and different program behaviors, and the method to vary $\text{RISK}_{\text{threshold}}$ will be investigated to make the redundancy determination best reflect the requirements of each workload.

This investigation will be introduced in the next chapter.

Chapter 5

Enhanced RazorProtector

In the previous chapter, we proposed RazorProtector [25] that uses an architectural redundant path as an alternative so as to help avoid the sharp performance degradation in traditional Razor-FF processors under high IR-drops. Assuming that a multi-issue processor is used to exploit parallelism, under an unexpected IR-drop zone, the processor will be dynamically reconfigured to use a redundant data-path to aid the recovery of timing errors. The redundant path works on the same instruction stream as the normal path. However, multi-cycle calculations are used in the redundant path, which guarantees a setup-error-free execution. Under a frequent timing error occurrence that is usually near the balancing point, the redundant path can help provide a very fast recovery. The paper [25] has also given a quantitative method to measure the criticality of operations to setup errors, as Delay Criticality Factor (DCF). By using a predefined threshold setup error risk level as RISK_{th}, RazorProtector can selectively apply the Redundant Mode to operations with long delay, so as to achieve a better power consumption than the conventional Razor-FF application.

However, the redundant data path gives visible performance draw back as it consumes resources for timing error coverage. As different applications will have different performance characteristics and distribution of instructions with different DCFs, it can be expected that the chances of applying the redundant mode of RazorProtector are highly depending on the program characteristics of the current workload. A statistically optimal, but static RISK_{th} may work efficiently for an average purpose, while may lose or even give worse efficiency in special

workloads. In this dissertation, we try a dynamic tuning method of controlling the application of parallel and redundant modes in RazorProtector according to the sampling data of current setup timing error rate, program operation vulnerability to timing errors, and the performance impact from redundant execution. Specifically, the most sensitive parameters to the final power efficiency, including the ILP (Instruction Level Parallelism), DCFs, and the loop duration, are tightly reflected into the dynamic control algorithm. The algorithm gives a tuned threshold $\mathrm{RISK_{th}}$ per each working set, which is then used to control the redundancy of the RazorProtector. The simulation results show that this approach can achieve 56% energy-delay-product (EDP) reduction, as compared to the traditional DVS and Razor-FF application.

Section 1 introduces the basic structure of the redundant data-path based RazorProtector and its working scheme to provide a fast setup error recovery. Section 2 gives the redundancy-adapting algorithm by using dynamic ILP and DCF studies. Section 3 presents the energy and performance study of the RazorProtector, compared with the conventional DVS application and static RazorProtector methods. Section 4 concludes the whole of this chapter.

1. Baseline method: RazorProtector

1.1 Basic Architecture

The redundant data-path structure in RazorProtector is illustrated in Fig. 5.1 (a). The whole architecture is composed of three pipelines in total: one primary pipeline (P-PIPE) and two supplemental redundant pipelines (R-PIPEs). The P-PIPE works under a normal setting and produces a fast calculation that is, however, vulnerable to setup timing error. Under a Redundant mode, the R-PIPEs are designed to use two cycles to finish each calculation in order to provide a timing error free execution. To detect timing errors under an aggressive voltage scaling, the pipeline registers follow the structure of a Razor-FF [9, 10, 11]. An error signal indicated by any Razor-FF will trigger a setup error recovery.

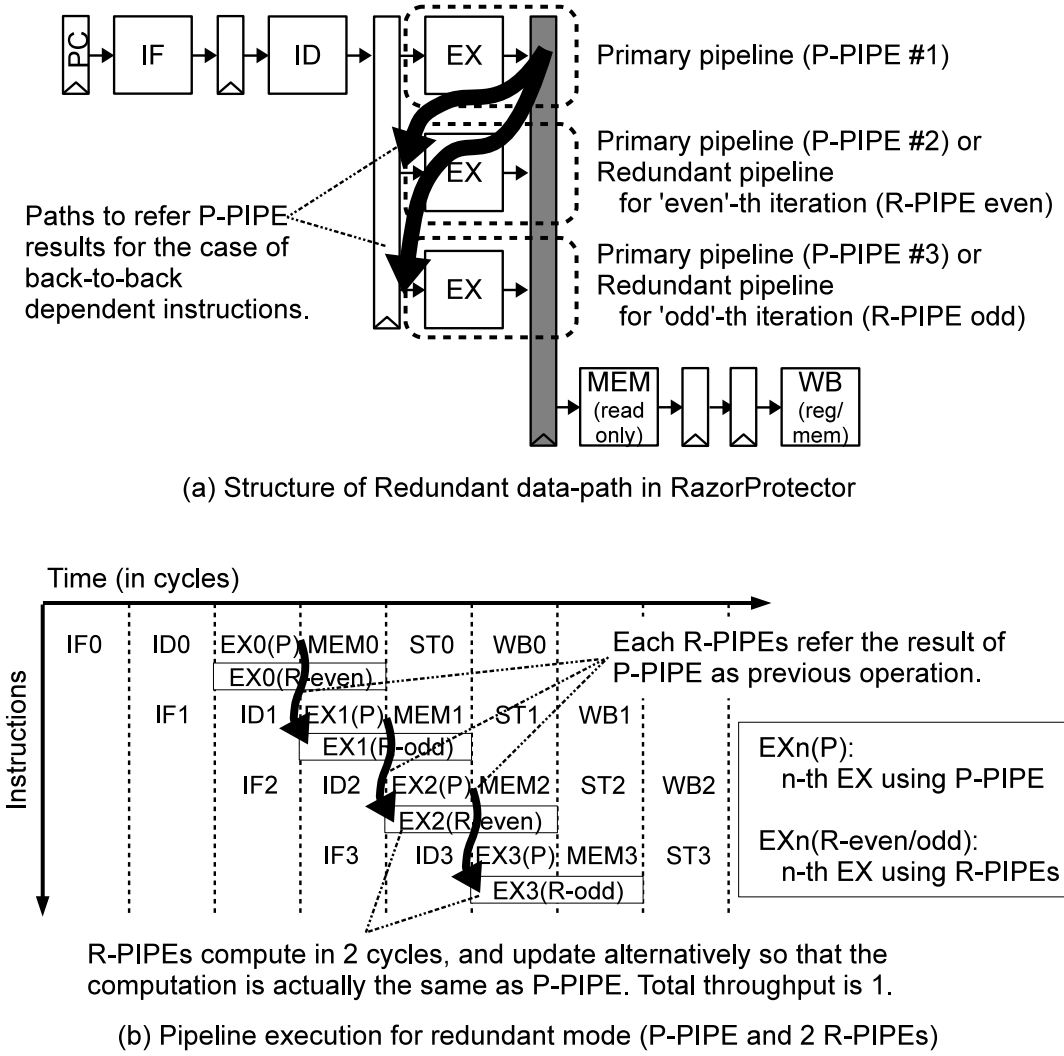


Figure 5.1. Redundant data-path system.

Fig. 5.1 (b) illustrates an execution example of the triple-pipeline architecture under the Redundant mode. Four instructions from 0 to 3 will be executed twice in this redundant data-path structure. More specifically, when instructions with even numbers, such as 0 and 2, are issued to P-PIPE, their executions will also be started on the R-PIPE1. In the succeeding cycle, odd-numbered instructions will be issued to both P-PIPE and R-PIPE2. Even when R-PIPE works under a halved throughput, the combination of two R-PIPEs can provide an additional execution of the four instructions. Note that the R-PIPEs require forwarding results of P-PIPE to finish the two-cycle-long calculation, when the two R-PIPEs are working on a dependent instruction pair [25].

When the Razor-FFs in the P-PIPE detect no error, its result can be used safely, and no recovery will be required. Fig. 5.2 shows the pipeline behaviors under a setup error occurrence. We show a comparison between a traditional non-redundant method and the redundant data-path method. Fig. 5.2 (a) is the case of the conventional method in the paper [9]. Assume that the setup error is detected by a Razor-FF in the execution stage 'EX1*'. The succeeding stages will be discarded, as 'flush' in Fig. 5.2 (a), before the re-execution of instruction 1 is started several cycles later. The penalty for recovery is same as a branch mis-prediction.

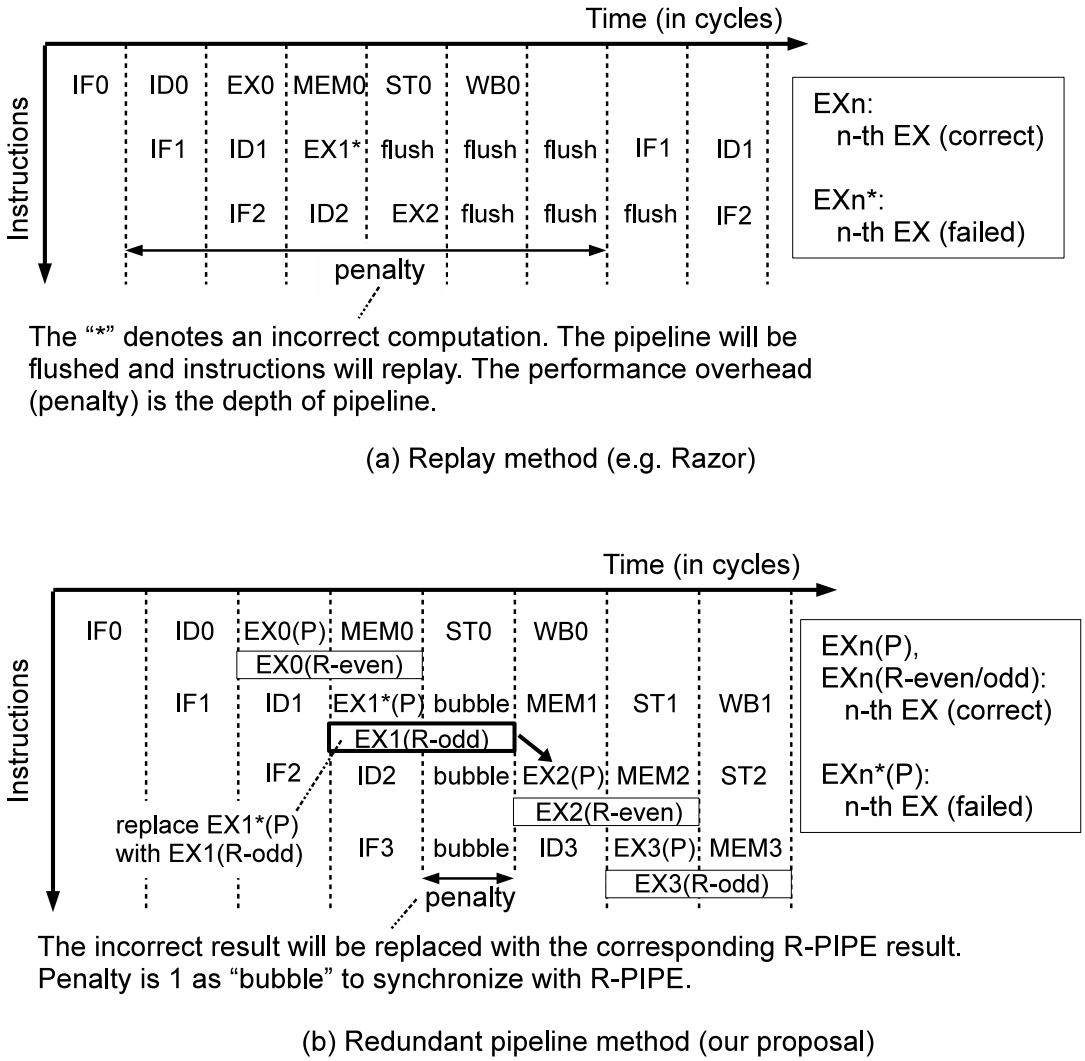


Figure 5.2. Penalty cycles in pipeline (conventional, proposal).

Alternatively, Fig. 5.2 (b) shows the case of recovery in the proposed method. After the timing error is detected in 'EX1*', although the output of the P-PIPE cannot be reliable, we can still use the EX1 (R-odd) output from R-PIPE after one cycle as the correct result. A one-cycle hazard, as 'bubble' in Fig. 5.2 (b) will be inserted to help take the correct value. Compared to the traditional Razor-FF usage, it is supposed that the redundant data-path will maintain some throughput even under a large timing error rate.

Apart from the Redundant mode, the three pipelines can use the same working frequency to process three different instructions at most per cycle, which represents a traditional Parallel mode. The Redundant mode trades computation resources for better timing error coverage. Thus, under a relatively large timing error rate, the Redundant mode must be used to achieve good energy efficiency.

This architecture is different from previous redundant architecture known as DIVA [12] and Slip-stream processor [13]. Both of these architectures have dynamic verification computation by a redundant core or thread. They are supposed to detect electronic errors that come from either transient or permanent faults. This research also focuses on setup error reduction, which may allow a greater timing fault tolerance than a normal Razor processor, especially when the error rate is high. A redundant data-path is established under Redundant mode by P-PIPE and R-PIPE. The error recovery is achieved by using the R-PIPE result when Razor-FF detects an error in P-PIPE. The connection in the architecture is tighter than DIVA architecture, so that only a one-cycle delay will be introduced under error detection. Recently, a new approach improves efficiency from Razor by predicting error and inserting an execution bubble speculatively [14]. Inserting an execution bubble creates enough time to avoid setup errors. Although this approach is similar to this research, it relies on speculative behavior and thus runs the risk of a large penalty. RazorProtector also uses prediction, but it can avoid a continuous penalty during Redundant mode.

1.2 Delay Criticality Factor (DCF)

As introduced in Section 1, the Redundant mode will only be required for operations of relatively long critical paths. For this purpose, we propose Delay

Criticality Factor (DCF) to measure risks related to the delay of operating units. The definition of DCF and risks of malfunction are described as follows:

$$\mathrm{Pr}_{\mathrm{path}} = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{t_{\mathrm{delay}} - t_{\mathrm{typdelay}}}{\sqrt{2\sigma^2}}\right)\right) \tag{5.1}$$

$$\mathrm{DCF} = \frac{\sum\left(\mathrm{Pr}_{\mathrm{path}} \times \mathrm{N}_{\mathrm{path}}\right)}{\#B} \tag{5.2}$$

$$\mathrm{RISK}_{\mathrm{setuperr}} = \mathrm{ERR}_{\mathrm{setup}} \times \mathrm{DCF} \tag{5.3}$$

(Notes: erf is error function)

Fig. 5.3 shows the relationship between logic structure and above equations. Operation unit A is corresponding to a certain DCF defined by Eq. 5.2. In general, a circuit has many paths from the start point to the end point in the propagation delay path. The dashed arrow indicates the propagation delay path corresponding to DCF of a certain operation.

In the above equations, $\mathrm{Pr}_{\mathrm{path}}$ is the probability of setup errors caused due to the delay of each critical path t_{delay}. In each operation unit (e.g. ALU in data-path), the $\mathrm{Pr}_{\mathrm{path}}$ of the activated path can be used to indicate its current setup error possibility. Equation 5.1 assumes that the delays from all critical paths follow a normal distribution around the average delay t_{typdelay}. According to Eq. 5.1, $\mathrm{Pr}_{\mathrm{path}}$ will be 100% in the case that t_{delay} is the same as the maximum delay in all critical paths. In Eq. 5.2, $\mathrm{N}_{\mathrm{path}}$ denotes the amount of nets included in each critical path. The $\mathrm{N}_{\mathrm{path}}$ serves as the weight of the corresponding path. $\#B$ is the amount of all nets included in the operating unit. The DCF is thus a weighted average gotten by combining the $\mathrm{Pr}_{\mathrm{path}}$ of each path. Each operation unit will have a corresponding DCF that measures its vulnerability to setup error due to the delay distributions of its circuit.

Table 5.1 is example of DCF for each instruction. This table also contains operation delay which is the average of t_{delay} corresponding to instruction. Each instruction has the DCF value respectively, it has a correlation with operation delay described as $\mathrm{Pr}_{\mathrm{path}}$. However, some instructions (e.g. ADD and ASR) have different correlation. It is caused by the difference of logic structure described as $\mathrm{N}_{\mathrm{path}}$ and $\#B$ in Eq. 5.2. In this chapter, the variation to the calculation delay from the input values has not been considered. The statistically DCF

in this dissertation gives a static analysis of the vulnerability measure of the operation itself. Adding the input data may reach a more accurate dynamic measure. However, the basic method of using DCF to tune a suitable adaptive use of RazorProtector is same for either a static or a dynamic DCF. Studying the impact from a dynamic input will be one of the future tasks.

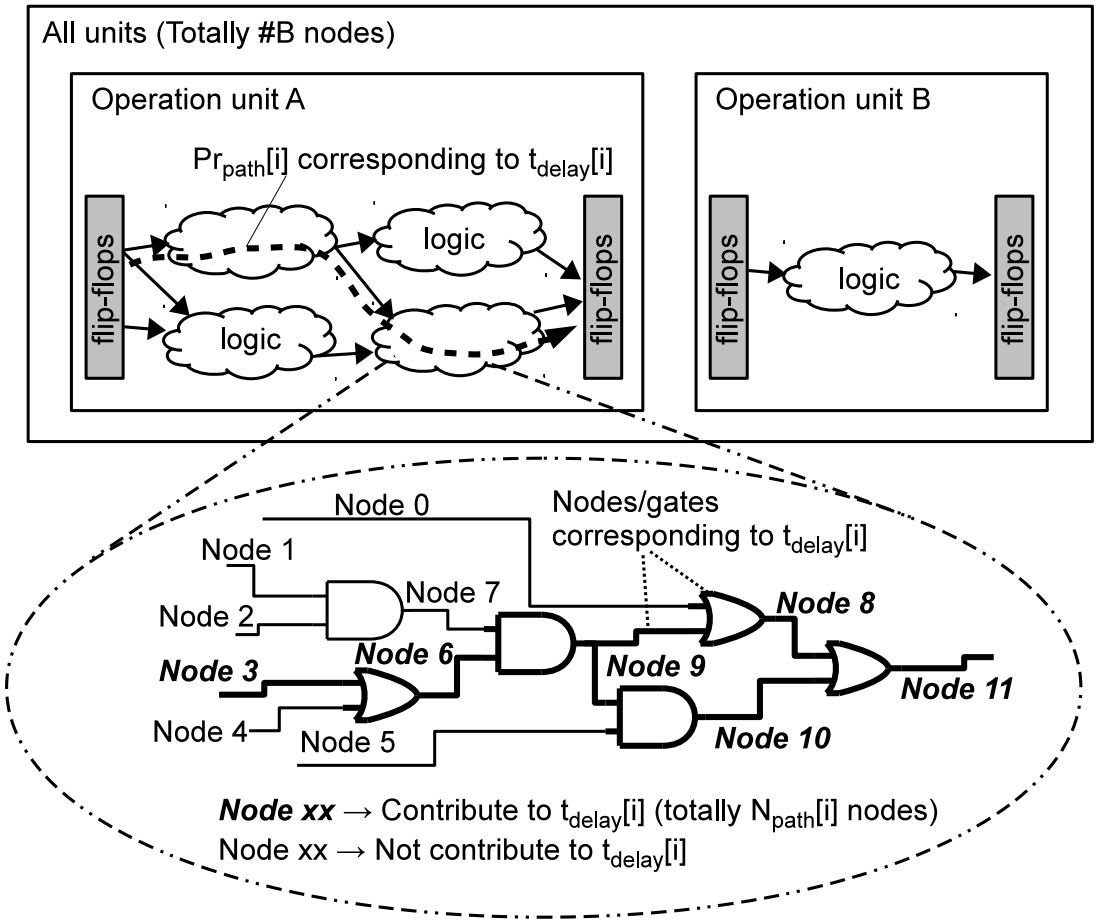


Figure 5.3. Relationship between logic structure and equation.

In Eq. 5.3, $\mathrm{ERR}_{\mathrm{setup}}$ is the setup error rate observed in the system (e.g. by error detection flip-flops). The risk of malfunction of each operation is denoted as $\mathrm{RISK}_{\mathrm{setuperr}}$, which is the product of the corresponding DCF of that operation unit and the $\mathrm{ERR}_{\mathrm{setup}}$.

As DCF determines the vulnerability of each operation to the setup error, it is possible to use this value to switch adaptively between Parallel and Redundant modes due to the detailed instruction flow. A DCF threshold is used to define the boundary in order to switch the mode, as in Eq. 5.4.

$$\mathrm{DCF}_{\mathrm{th}} = \frac{\mathrm{RISK}_{\mathrm{th}}}{\mathrm{ERR}_{\mathrm{setup}}} \tag{5.4}$$

Table 5.1. DCF for each instruction.

Instruction	Operation delay	DCF	Functionality
MUL_ADD	73.8	30.4	Multiply and Add
MUL	46.1	22.8	Multiply
ADD	34.8	17.4	Add
ASR	45.5	16.4	Arithmetic shift

(Notes: 100 is maximum at each value.)

2. Scheme to adapting redundancy

2.1 Program parameters to select P_mode or R_mode in RazorProtector

The Parallel mode (P_mode) in RazorProtector uses the three pipelines to multi-issue instructions to exploit ILP, while the Redundant mode (R_mode) uses the two-cycle error free execution in partial pipelines to largely reduce the error recovery penalty. We use Figure 5.4 to show the applicability of either P_mode and R_mode by comparing RazorProtector to the traditional DVS method. The data in this figure is based on a trace-based simulation. The x-axes in these figures are the supply voltage values, normalized by the optimally balanced voltage V_{opt}. We assume that an unexpected IR-drop will make the supply voltage go from 1.0 toward 0.9. The corresponding setup error rate during this period is calculated by following the alpha-power law delay model [15, 16]. The y-axes show the normalized energy consumption.

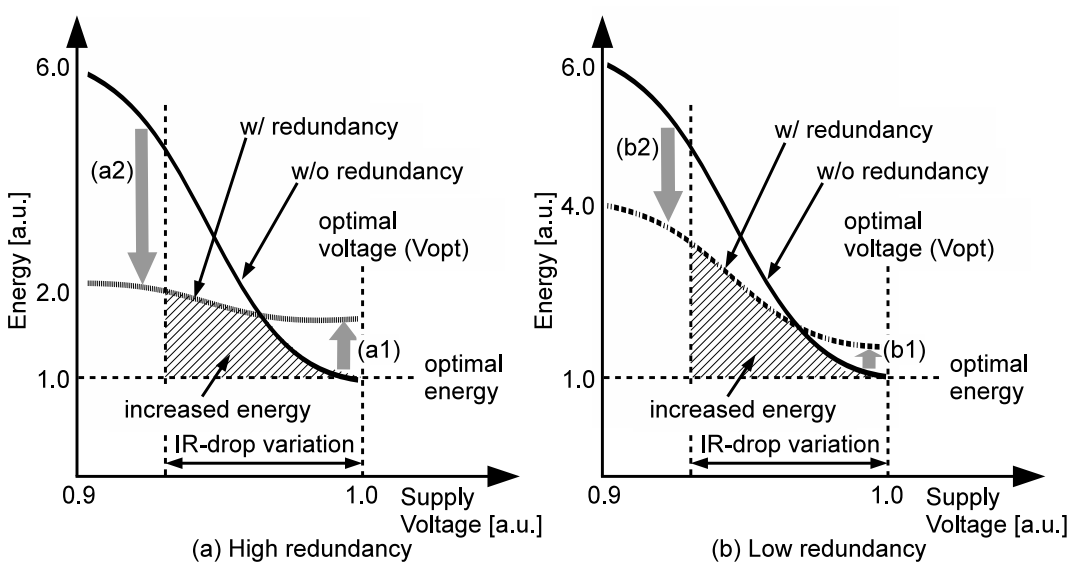


Figure 5.4. Increase of energy consumption due to IR-drop.

It can be easily observed from Fig. 5.4 that the high redundancy has a much larger reduction of energy consumption in the high error zone, as comparing the arrows of (a2) and (b2). This is obvious because R_mode only has a one-cycle recovery penalty and is thus preferred under conditions where recovery is frequently required.

The difficult turning point is at the low error rate part. As given in Fig. 5.4, the redundant data-path will have a higher energy consumption value than the traditional method at regions near V_{opt}, where the error rate starts to add some visible impact while the impact is not high enough to tune the performance loss in the R_mode.

As considering the energy-delay-product (EDP) as a measure, the P_mode and R_mode will have the following balancing point under a given error rate is:

$$\frac{\text{EDP}(\text{P}_{\text{mode}})}{\text{EDP}(\text{R}_{\text{mode}})} = \frac{\text{N}_\text{I}/\text{IPC}_{\text{P}_{\text{mode}}} + \#\_\text{of\_errors} \times \text{n}_{\text{depth}}}{\text{N}_\text{I}/1 + \#\_\text{of\_errors} \times 1} \tag{5.5}$$

Here, N_I is the total number of instructions and #_of_errors is the number of errors. Note that #_of_errors is the number of visible errors where the large DCF of instructions makes data arrive later than the setup requirement. It actually measures the average DCF of this workload. $\text{IPC}_{\text{P}_{\text{mode}}}$ is the ILP measure of this workload under P_mode. The n_{depth} is the recovery penalty under P_mode, which is same to the pipeline flush penalty. According to R_mode working mode, the $\text{IPC}_{\text{P}_{\text{mode}}}$ has a constant value of 1. Its recovery penalty, however, is also 1 cycle.

Under some extreme case where #_of_errors $= 0$, the R_mode has similar energy efficiency as P_mode only when $\text{IPC}_{\text{P}_{\text{mode}}} = 1$, which indicates a very low ILP workload. When $\text{n}_{\text{depth}} = 5$ and 12.5% of the instructions are with very long DCF to cause faults, R_mode should has an $\text{IPC}_{\text{P}_{\text{mode}}} > 2$ to be better than R_mode. According to these very rough calculations, it is possible to use performance counter to measure both ILP and the number of visible errors to give an estimation of the successive working mode.

2.2 Algorithm to tune the adaptive redundancy

In this section, we are using architectural method to give a dynamic redundancy control of the RazorProtector. Figure 5.5 shows a typical change of ILP and DCF

along the time-line in these benchmarks. It can be easily observed that applications composed of hot loops that give recursive program characteristics. Both ILP and DCF will be stable for a long time and then shift to other values after a sudden change. Accordingly, Figure 5.6 shows enhanced control architecture.

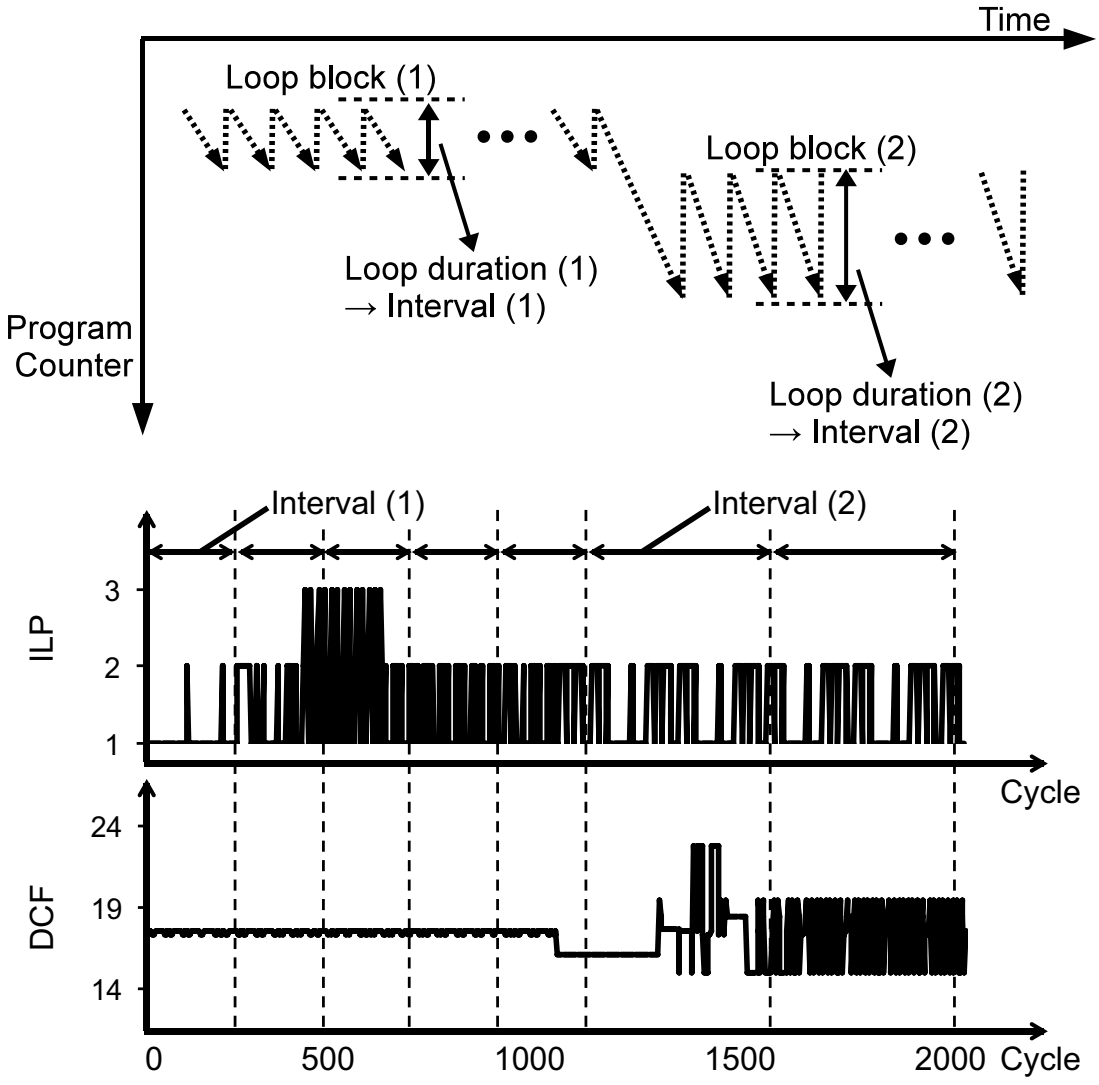


Figure 5.5. Relationship of the loop duration and the required interval for updating $\mathrm{RISK_{th}}$.

"A" in Fig. 5.6 is the decoding phase of the processor, where we can get the DCF of the current instruction according to its operation type. This can be easily achieved by preparing a DCF lookup table inside the ID stage which is indexed by the operation type [25]. The DCF of this pending instruction will be compared to the threshold DCF_{th} to determine the suitable redundancy level, as P_mode (parallel, non-redundancy) and R_mode (redundant data-path). As introduced in Section 2.1, DCF_{th} should be tuned to fit for the program characteristics and error rate to achieve an optimal energy efficiency by using the RazorProtector. In the architecture shown in Fig. 5.6, we use an error rate sampler to gather the error detection signals generated from error detecting flip-flops, as (B) in Fig. 5.6. The value of ERR_{setup} is then calculated according to the number of collected errors in the sampling period. Accordingly, it is possible to predict and tune a suitable $RISK_{th}$, and DCF_{th} can then be easily given with the help of Eq. 5.4.

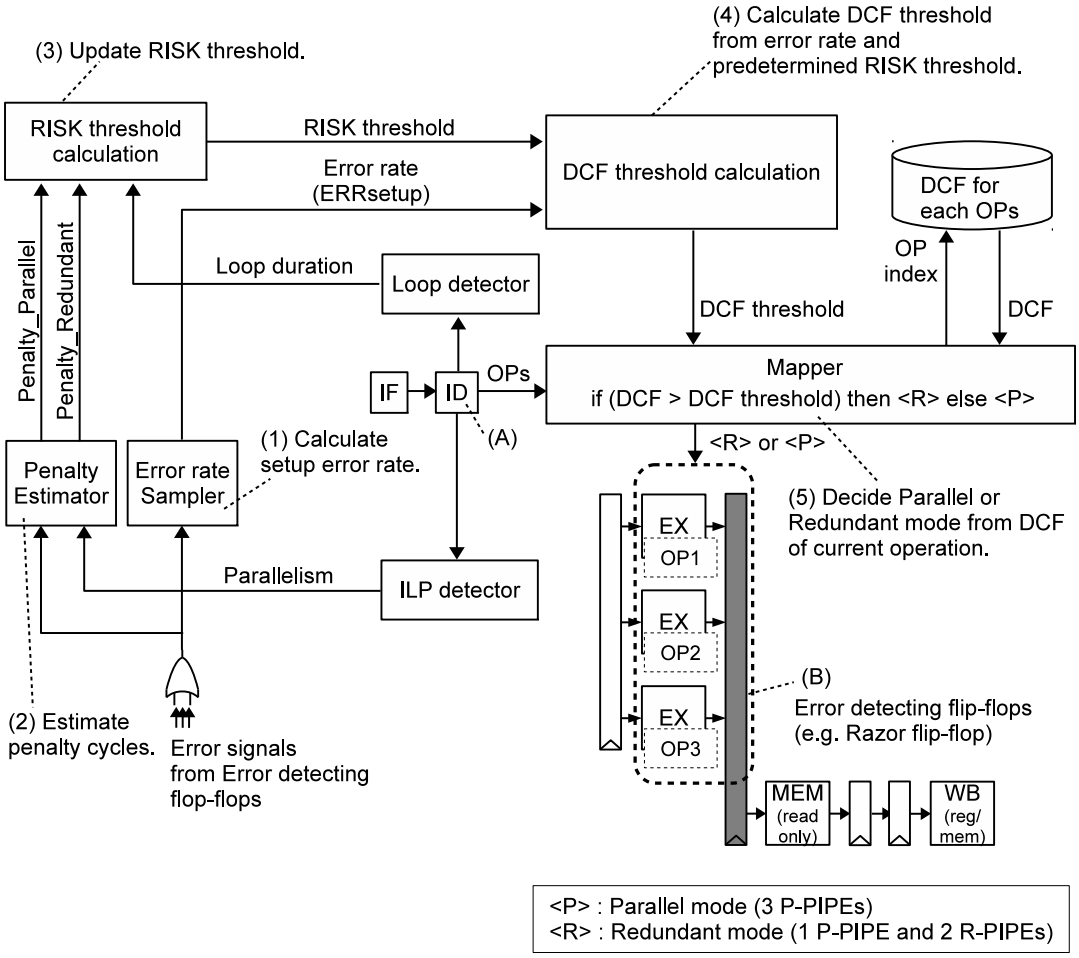


Figure 5.6. Determine redundancy by DCF, $\mathrm{ERR_{setup}}$ and $\mathrm{RISK_{th}}$.

Figure 5.7 shows the detailed algorithm that we used to tune a suitable RISK_{th} for program hot loops. The algorithm is written in a style of processor simulator. The actual processing in the hardware is, however, working simultaneously. As shown in Fig. 5.7, at each decoding stage (Fig. 5.6 (A)), the decoder will give the DCF of the current instruction group and RazorProtector can choose from P_mode and R_mode according to the DCF of the pending instruction group and the tuned DCF_{th}. The selected P_mode or R_mode will be used for the execution of this instruction group. Note that under P_mode, the processor is a multi-issue processor, which supports three issues at most. Under R_mode, the instruction will be put into the pair of P_PIPE and R_PIPE1 or the pair of P_PIPE and R_PIPE2 to guarantee a setup-error-free execution.

```
enum {P_mode, R_mode} curr_mode;
int xscore[]; /* Scores of each mode */

while (1) { /* cycle */
    inst[] = fetch_inst();
    /* Fig. 5.6 (A) */
    curr_mode = (DCF > DCF_th) ?  R_mode :  P_mode;

    berror = exec(inst[], curr_mode); /* Fig. 5.6 (B) */
    /* Fig. 5.6 (2) */
    if (berror) {
        recover(inst[], curr_mode);
        xscore[P_mode] += n_depth × IPC × weight;
        xscore[R_mode] += 1;
    }
    else if (curr_mode == R_mode)
        xscore[R_mode] += IPC - 1;

    if (++n > sample_interval) {
        /* Reconfigure threshold per interval */
        /* Fig. 5.6 (3) */
        if (xscore[P_mode] > xscore[R_mode])
            RISK_th -= STEP;
        else
            RISK_th += STEP;
        n = 0;
        xscore[P_mode] = xscore[R_mode] = 0;

        DCF_th = RISK_th / ERR_setup; /* Fig. 5.6 (4) */
    } /* end of reconfiguration */
} /* end of this cycle */
```

Figure 5.7. Tuning algorithm to find optimized $\mathrm{DCF_{th}}$ and setting redundancy mode.

The signal "berror" in Fig. 5.7 is then used to indicate whether there is setup error in the execution of the three pipelines under P_mode, or in the execution of P_PIPE under R_mode. When there is a setup error, "xscore" will be increased to represent the penalties of the recovery costs of both modes. P_mode requires a pipeline flush, and accordingly it loss of instruction issue chance is IPC $\times$ n_{depth}, with weighting factor. The value n_{depth} is the pipeline depth, which connects to the flush penalty in a normal pipeline. R_mode has a much smaller recovery cost from the design of the RazorProtector. It can forward the data from R_PIPE back to the P_PIPE to achieve a one-cycle setup error recovery. Therefore, its "xscore" is increment is 1 (Fig. 5.7). However, when there is no "berror" in the P_PIPE execution, the loss for the R_mode is the chance of multiple issue, which is IPC - 1 in the architecture.

After the accumulation of both "xscores" in the above algorithm block for a sample period, the sampled "xscores" are used to give an estimation of the $RISK_{th}$. Here the sample period can be set as the loop body length of the studied hot loop. The length of the period can be easily extracted from the information contained in the loop exit instruction, or the backward short jump instruction. Both can be analyzed in the ID stage. According to the result of the "xscores" comparison, $RISK_{th}$ will be increased when the penalty of R_mode is higher and vice versa. Under a same ERR_{setup}, a smaller $RISK_{th}$ will result in a smaller DCF, which will further give a tendency of more applications of R_mode.

The "STEP" affects the time to re-tune the balancing point of $RISK_{th}$, and it also affects the granularity of re-tuning. Both the time to re-tune and the granularity of re-tuning are trade-offs to set. In this dissertation, $RISK_{th}$ varies from 0.01% to 0.1% and then to 1% respectively. In the first range, the STEP is 0.01% while it becomes 0.1% in the second 0.1% to 1% range.

By using this method, we have successfully reflected the average IPC into the $RISK_{th}$. When the IPC of the workload is high, R_mode gets more "xscore" increase of non-erroneous execution. In other words, if IPC is very low, there is almost no performance gaining of P_mode, so that R_mode will be more preferred, achieved by a slowly incremented "xscore[R_mode]". Beside the ILP, the average DCF of the workload is reflected into $RISK_{th}$ by the condition block of "berror" in Fig. 5.7. A higher average DCF will have a larger chance to enter the "berror

== true" block and P_mode will get more penalty, which finally results into a decreasing tendency in DCF_{th}. Therefore, from the above two points, we have included all possible tuning parameters, as ILP, DCF, and ERR_{setup} by the help of this relatively easy-to-implement algorithm.

As the proposed algorithm uses simple additions, and comparisons to give a quantitative evaluation of potential harmfulness of the setup error, the additional hardware for the controlling method can be controlled under a small level. The major hardware to implement this algorithm is in the DCF evaluation part in the decoding phase, where a lookup table is required to give the DCF value according to the operation type. The instruction set architecture (ISA) shows that a 256-entry table is sufficient for the DCF purpose. In addition, the $DCF_{th} = RISK_{th}/ERRsetup$ will also take hardware. This equation needs division, but it can implement simple hardware with addition and shift. The delay of this division is tolerable as it is only required at a less frequent interval granularity. In total, the evaluation results indicate that these additional logics will introduce a 1.6% increase and cause 0.5% additional power for logics. The additional memories will introduce a 1.0% increase, which causes 1.1% additional power for memories.

This method is based on the study and tuning result of program characteristics DCF in a "sample_interval". It is possible to be applying this method to any platform with basic locality. Even for a many-core platform which run many thread simultaneously, this method is expected to tune the core to the best $RISK_{th}$ according to the thread that occupies the CPU core currently. The study of the platform without any locality, i.e., very fine-grained context switch, is out of the scope of the dissertation.

3. Practically Simulated EDP results

In this section, we introduce the effectiveness of the proposed RazorProtector with the adaptive redundancy from the tuning method, under possible large IR-drop zones. We use the Energy-Delay-Product (EDP) measure for the efficiency study, assuming that the preferred platform is a workstation or mobile devices and so on, where EDP applies best. The data are collected from a cycle accurate simulator, which contains performance simulation and power estimation based on

a mathematical model including the alpha-power law delay model [25, 15, 16].

The evaluation framework had been proposed in the paper [25], which is based on a cycle accurate processor simulator. The simulation platform is described in detail in Table 5.2. The corresponding circuit data is extracted from a special FR-V processor [21]. The voltage will be scaled within a range from 0.8V to 1.3V in this simulation. The voltage scaling algorithm has following steps.

1. The voltage down-scales when there is not error.
2. Keeps when error rate is under a tolerable value.
3. Up-scales when error rate is intolerably high. The up-scaling voltage step corresponds to "Voltage scaling slope" in Fig. 5.8.

Table 5.2. Parameters of cycle-based simulator.

Instruction issue width	4/cycle
No. of General purpose register (GR)	32
No. of Media register (MR)	32
L2 ⇒ L1 transferring	8bytes/cycle
Instruction cache	4way 16KB (64bytes/line)
L1 cache	4way 16KB (64bytes/line)
L1 ⇒ L0 transferring	12bytes/cycle
Store buffer	4entry

3.1 Effectiveness of the $\mathbf{RISK_{th}}$ adaptation

The main parameters that we used to give an optimized control of redundant data-path application is based on IPC and DCF, as well as the current sampled setup error rate ERR_{setup}. In real processor using DVS method, another impacting factor is the voltage changing speed. After the IR-drop happens, the timing error rate will remain at a relatively high value before the voltage is re-adapted to the balanced level. Therefore, the voltage scaling up speed directly affects the working model of RazorProtector. In some extreme cases, if the voltage scaling up penalty is 0, the traditional Razor-FF processor will have the best EDP, since the

re-adapting can be finished in the next cycle. If voltage scaling is extremely slow, the redundant mode should always be applied by setting a near-0 risk. However, this speed is more related to the DVS technique in processors, which cannot be directly obtained by the RazorProtector method. In this section, we explore the effectiveness of the enhanced RazorProtector by studying several possible voltage scaling speeds.

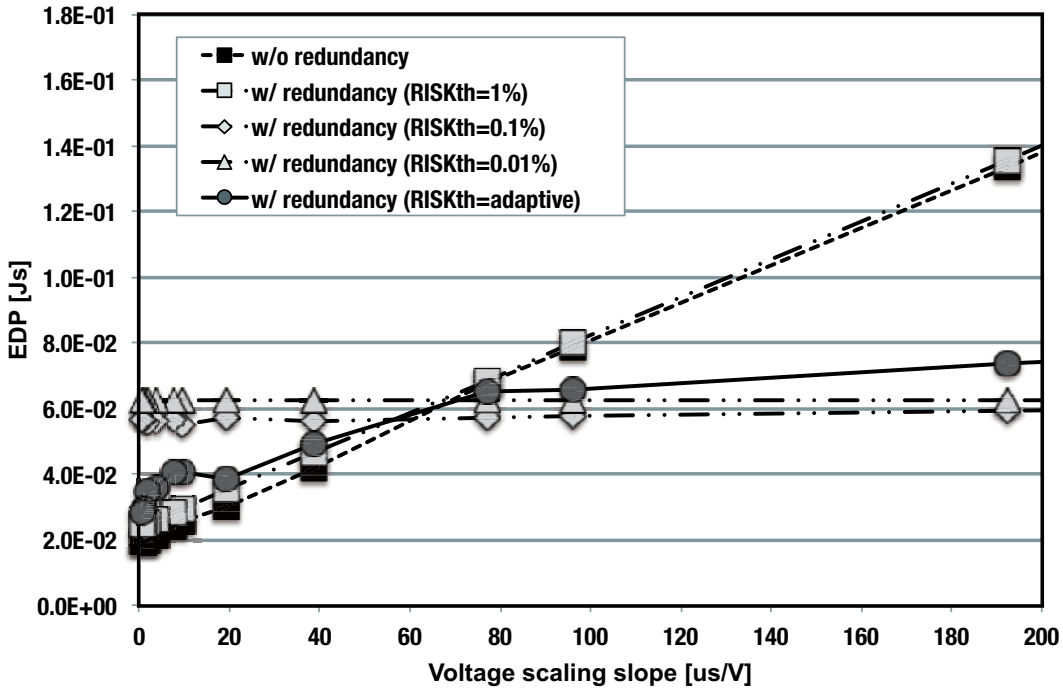


Figure 5.8. EDP results of different $\mathrm{RISK_{th}}$ (unsharp, 10% IR-drop).

We give the EDP result of the benchmark *unsharp* in Fig. 5.8, as the representative workload to illustrate the results of non-RazorProtector, static and enhanced RazorProtector. Beside the traditional non-redundant data-path DVS utilization, three static RISK_{th} values, as 0.01%, 0.1% and 1%, are tried for the comparison purpose. The IR-drop in these experiments is set as 10% to the maximum voltage, representing a relatively high setup error injection rate. The EDP results of the corresponding execution of *unsharp* are shown along the vertical axis in Fig. 5.8. The voltage scaling speed is given in the horizontal axis, which is the time duration used in each unit voltage scaling. The left most value $0\mu\text{s/V}$ is the most ideal voltage scaling situation, while the rightmost value represents a relatively practical scaling in current processors. Note that the redundancy mode is used only before the voltage balancing point is re-adapted. After that, normal multi-issue parallel execution will take position under the balanced voltage V_{opt}.

Basically, RISK_{th} 0.01% is similar to 0.1% and is better than both the traditional DVS and the static RISK_{th} 1% applications under the low voltage scaling speed zones. The traditional DVS method works with no redundancy aids and can be expected to have a very low IPC during the voltage re-adaption, because of frequent long hazards of setup time errors. The static RISK_{th} 1% application leads to a relative preference to P_mode than R_mode, which may miss some parts of possible application of the redundancy mode in the low voltage scaling speed zones. However, the results become different in the high voltage scaling speed zones. Even the 100% parallel mode without redundancy outperforms RISK_{th} 0.01% and 0.1% when the voltage scaling can be done quickly. This may come from the large IPC of *unsharp* which is around 2.07. When the voltage scaling is done without large penalty, P_mode is more preferred to fully exploit ILP for a reduction of EDP. These observations also emphasize the necessity of an adaptive RISK_{th}.

The EDP results of the dynamically adapted RISK_{th} by the tuning method given in Section 2.2 clearly demonstrate the effectiveness by taking the inflection point at the crossing of RISK_{th} 0.01%, 0.1% and 1%. The EDP results of each voltage scaling speed are almost the optimized ones from all these dashed lines. It indicates that by using this tuning method, the program execution has been properly duplicated, and the redundant data-path provides a good covering of

the recovery. Accordingly, even though the processor stays under an insufficient voltage longer when the voltage scaling speed is low, the performance will not be damaged because the flush-based recovery has been avoided by the redundant data-path at those critical instructions. This dissertation uses EDP as the metric to show the efficiency of processing workload. As described in Section 3, the draw back of Razor is taking the time to re-tune the balancing point when it causes setup-error. Considering this delay, EDP is a good energy efficiency measure for low-power designs as it takes both energy consumption and performance into account. It is specially suitable for platforms including laptop and smart-phones where energy efficiency is most preferred. An EDP reduction can be achieved by two ways: 1. low power consumption in a given deadline performance. 2. same power consumption but fast completion of the same task. In this research, we are trying more on the second meaning by keeping the already low power consumption processor with Razor FF experiencing similar performance even under large IR-drop zones.

3.2 EDP reduction results

This section gives a summary of the EDP reduction that RazorProtector can achieve, by applying it onto the following benchmarks. In this dissertation, the traditional Razor FF [9] without any pipeline redundancy serves as the baseline processor. This chapter then compares with a previous research RazorProtector in [25], which uses a fixed $\mathrm{RISK_{th}}$. Note that both EDPs of the previous static RazorProtector and this dynamic one are normalized by the baseline traditional unprotected Razor use.

We use 4 filtering functions *unsharp*, *blur*, *FI-a* and *FI-b* from image processing programs. Function *unsharp* is the same program introduced in Section 3.1. *FI-a* and *FI-b* are parts of a frame interpolation, where *FI-a* is searching block corresponding to minimum SAD, which mainly contains comparison instructions. *FI-b* is interpolating pixels corresponding to searching results, where address calculations and memory copies are the top used operations. *blur* is blurring filter, whose main instructions are additions, shifts and multiplications.

We also use 6 benchmark programs *basicmath*, *qsort*, *susan*, *patricia*, *sha* and *jpeg* from MiBench[26]. These programs cover the Automotive and Industrial

Control category (*basicmath*, *qsort*, *susan*), the Network category (*patricia*), Security (*sha*) and the Consumer devices category (*jpeg*). These benchmarks make up the various workloads to study the final EDP reduction results of the RISK_{th} adapting RazorProtector approach.

The EDP results of these benchmarks are shown in Fig. 5.9, as normalized by the EDP of traditional DVS and Razor-FF technique, which is already known as an effective power saving method under low IR-drop zones. A practical voltage scaling speed of 100μs/V has been used in these executions. This result shows that the RazorProtector method can contribute EDP reduction to all the application. In average, about 75% EDP reduction can be achieved by applying the dynamically adapted RISK_{th}. It indicates that RazorProtector can be used to maintain the applicability of DVS even when the unexpected IR-drop reaches a relatively high level.

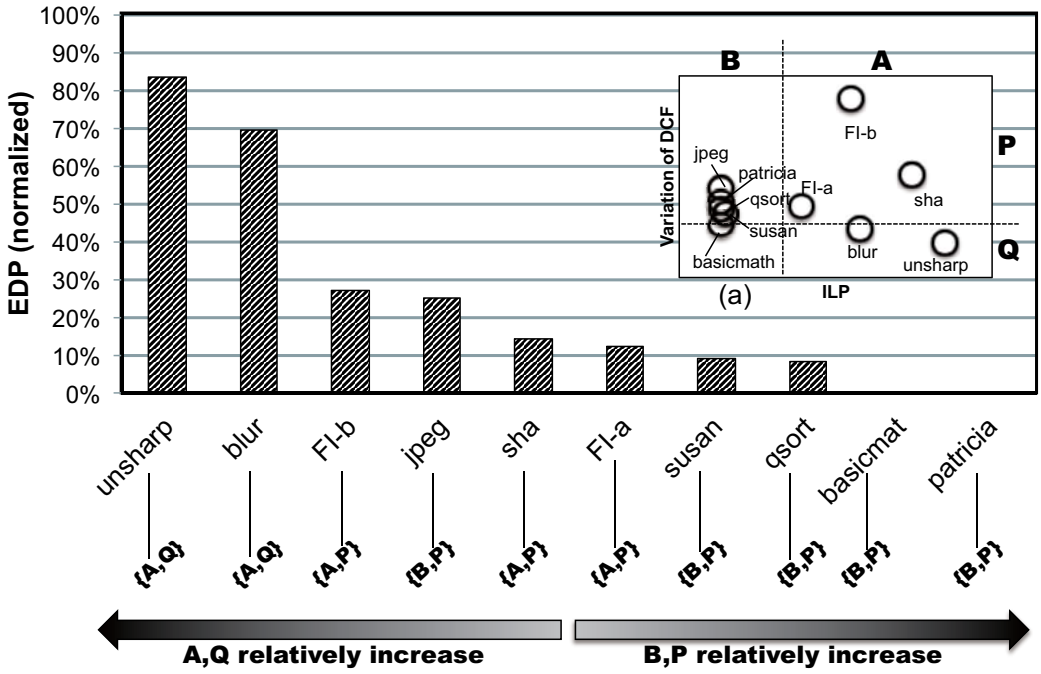


Figure 5.9. EDP reduction in various applications.

It can also be observed from Fig. 5.9 that the ability of furthering EDP reduction by the enhanced RazorProtector is largely varying in workloads. The benchmarks are listed in Fig. 5.9, following a decreasing EDP order. Among all the benchmarks, the enhanced RazorProtector gets 17% reduction in *unsharp*, but achieves near 100% reduction in *basicmath* and *patricia*. As has been introduced in Section 2.1, the applicability of R_mode in RazorProtector connects to the program characteristics ILP and DCF. The sub-figure in the up-right corner, Fig. 5.9 (a) gives a rough analysis of these two characteristics of all the benchmarks.

Specifically, the horizontal direction of Fig. 5.9 (a) gives the IPC difference. Benchmarks in Group A have larger IPCs than those in Group B. The vertical direction in the subfigure demonstrates the variation of DCFs in these benchmarks, calculated as the standard deviation of DCF. Benchmarks in Group P show more deviations than Group Q. A large DCF variation indicates that DCF in the workload varies a lot. A balanced voltage may be good for some instructions, but can cause relative more setup errors when other instructions of larger DCFs are in execution. Therefore, it is possible to find more application chances dynamically to enable/disable the redundant data-path in Group P.

The EDP results have proven these assumptions that benchmarks in Group B and P can have more EDP reductions than Group A and Q. Accordingly, the most EDP reductions have been achieved in Group (B,P) and vice versa. The EDP efficiencies of benchmarks in Group (A,P) are between the other two groups, while no benchmarks in the workloads fall into Group (B,Q). Therefore, this adaptation algorithm in Section 2.2 correctly recognizes the most suitable program characteristics to the setup errors. It can thus be regarded as an accurate dynamic algorithm for the purpose of RazorProtector application.

Fig. 5.10 gives further comparison between static and enhanced RazorProtector applications, under a voltage scaling speed of 100μ/V. Due to the relatively slow voltage scaling speed, low RISK_{th} 0.01% and 0.1% are better preferred than RISK_{th} 1% to avoid P_mode in high DCF zones. Comparatively, all RISK_{th}s do not work as efficiently for unsharp and blur, due to this program characteristics (A, Q), as in Fig. 5.9. Among all, this method can successfully give an EDP near to the statistically good RISK_{th} 0.01% and 0.1%. Only in *jpeg*, the result

is visibly worse than others. This may be because the characteristics in *jpeg* do not help the algorithm give a clear difference in P_mode and R_mode. The slow 100μs/V voltage changing speed actually requires a more preference of R_mode, which may thus cause a difference in the real application and algorithm determination. However, as discussed above, the speed value is not an obtainable one in the algorithm. Circuit level method is required to help this algorithm. In average, about 75% EDP reduction can be achieved by applying the dynamically adapted RISK_{th} at this condition. Compared to the statically best RISK_{th} 0.01%, this method still achieves 92% efficiency, given by $\frac{\text{EDP}_{\text{RISKth=0.01\%}}}{\text{EDP}_{\text{RISKth=adaptive}}}$ in reducing the EDP.

Fig. 5.11 shows EDP comparison under a practical voltage scaling speed as 40μs/V. Under this relatively medium voltage scaling speed, the P_mode and R_mode become equally preferred by the program, i.e. RISK_{th} 0.01% is no longer solely give the best EDP. For (A,Q) programs unsharp and blur, the RISK_{th} 1% performs better than others. The adaptive RISK_{th} given in this tuning algorithm accurately presents a selection between these static thresholds. Finally, the average EDP gives best EDP reduction 56%, normalized by the traditional DVS application.

However, the EDP results of "jpeg" becomes worse by applying the adaptive tuning, as in Fig. 5.11. A further analysis reveals that the large variation of jpeg impedes the turning of DCF and RISK_{th}. Generally, the basic idea in this dissertation is adapting RISK_{th} by the recurrence of loop iterations. We are focusing on how to reflect the DCF and current error rate into the RISK_{th}. Fig. 5.12 gives the analysis results of the variation of DCF and loop size. If the number of loop iterations varies a lot, such as in "jpeg" (A), the RISK_{th} can not get a very stable tuning result. Under this situation, applying fixed RISK_{th} will be better. This additional stable or unstable program characteristic can be detected by basic block based categorizing, working set signature analysis, and so on, which is not the scope of this dissertation. However, the method to use an adaptive RISK_{th} can be orthogonally applied with other method like signature. For example, working set signature can detect recurring intervals[27], and in each interval, we can apply a tuned RISK_{th}, which may thus achieve best efficiency. The difference case to jpeg is FI-b (B), which gets a largely varied DCF feature but relatively stable

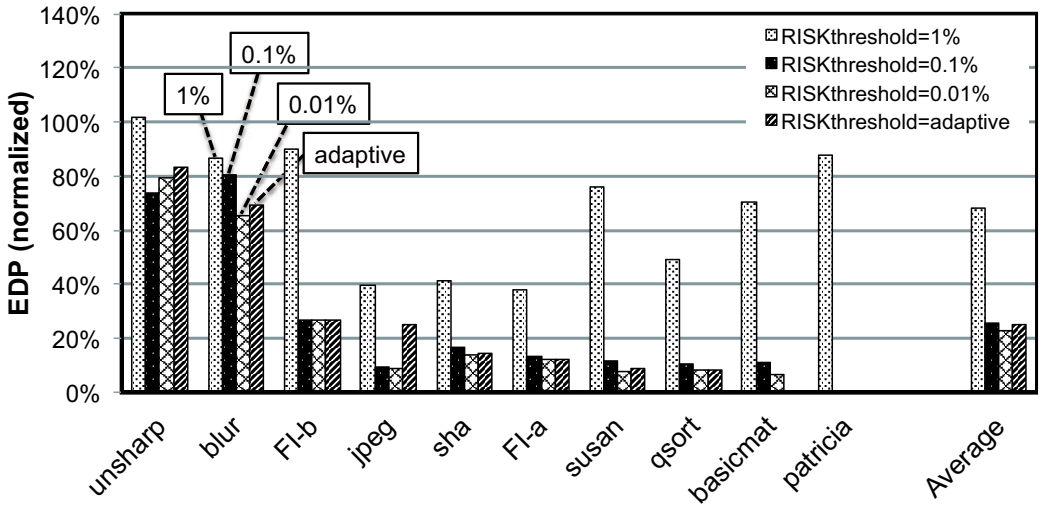


Figure 5.10. EDP comparison between fixed and adaptive $\mathrm{RISK_{th}}$. (100μs/V voltage scaling slope)

program phases, by showing a stable number of loop iterations. Together with the finding that the adaptive method achieves a good result in FI-b, it indicates that the method can get correct DCF tuning when other program characteristics provide enough locality and stability.

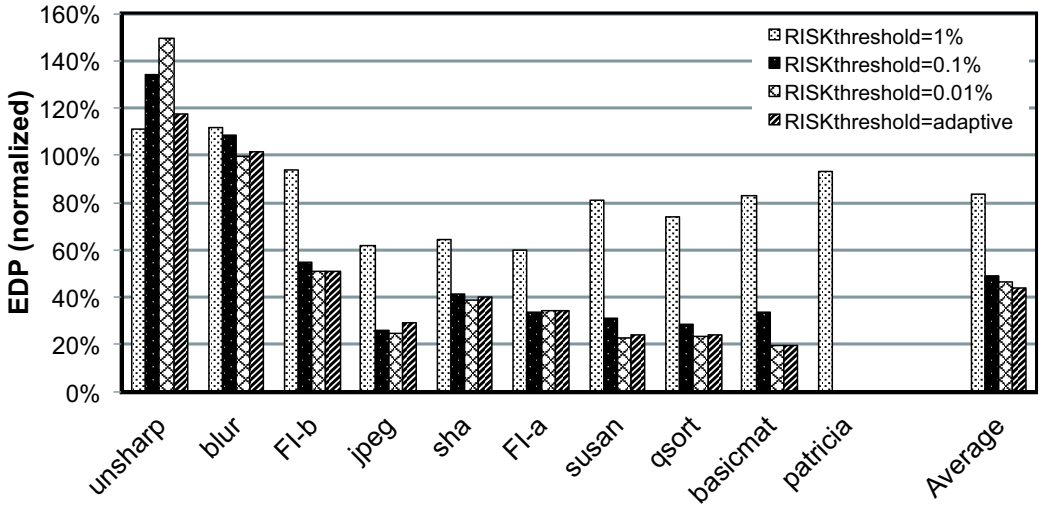


Figure 5.11. EDP comparison between fixed and adaptive $\mathrm{RISK_{th}}$. (40μs/V voltage scaling slope)

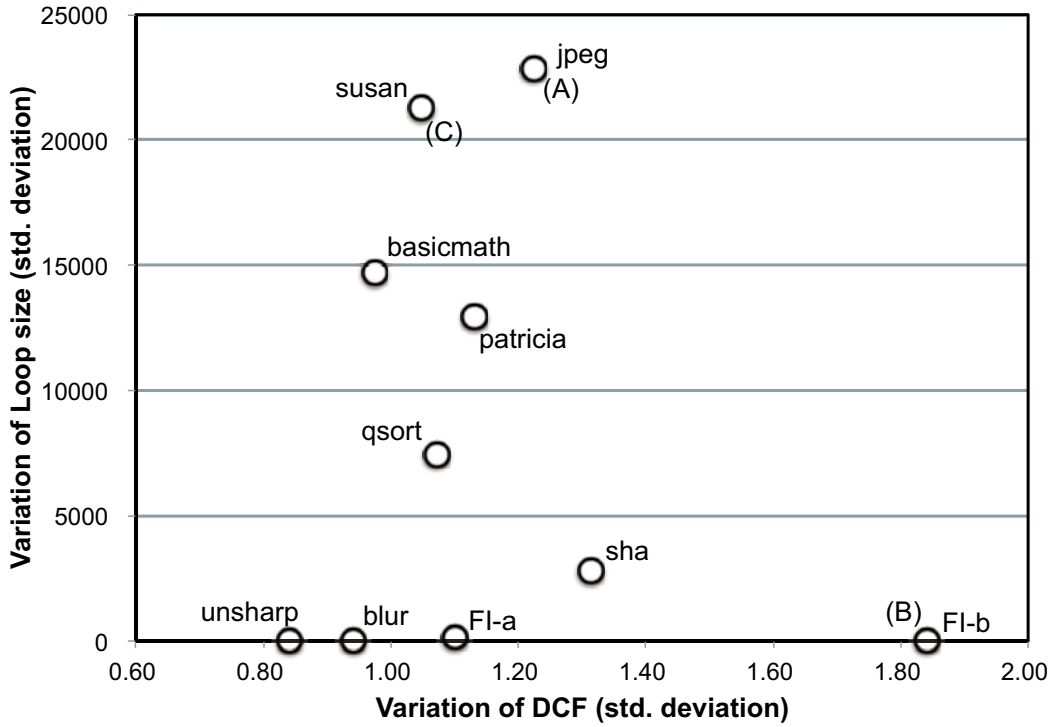


Figure 5.12. Application analysis for variation of DCF and Loop size.

4. Conclusions of this Study

This chapter proposed enhanced RazorProtector, a redundant data-path based method to help reduce the recovery penalty for processors in which DVS is aggressively applied. A program characteristic based adaptive redundancy was used to best tolerate unexpected IR-drops at post voltage balancing regions, assuming to use a special metric DCF to measure the setup error vulnerability. The results show that RazorProtector can help maintain Razor energy efficiency for a processor with a microsecond order voltage scaling restriction.

We evaluate EDP reduction for various applications from image processing programs and MiBench[26], under a practical voltage scaling speeds as 100μs/V and 40μs/V. Under a medium scaling speed 40uV/s, the adaptive method shows its efficiency by outperforming all static controls. In summary, 56% EDP reduction can be achieved by this method as compared to traditional DVS application, under high IR-drop zones.

Chapter 6

Conclusion

This dissertation proposed a redundant data-path based method to help reduce the recovery penalty for processors in which DVS is aggressively applied.

Firstly, we preliminarily studied the feasibility of redundant data-path and the risk metric to control redundancy. An adaptive redundancy was used to best tolerate unexpected IR-drops at post voltage balancing regions, assuming to use a special metric DCF to measure the setup error vulnerability. The results show that under a large setup rate caused by a 10% unwanted voltage drop, the adaptive redundancy can reduce energy consumption up to 52% and performance loss up to 12%.

Secondarily, we assembled the idea as the RazorProtector, a redundant data-path based method to help reduce the recovery penalty for processors in which DVS is aggressively applied. The results show that under a large setup rate caused by a 10% unwanted voltage drop, the adaptive redundancy can reduce EDP up to 78% at the 100μs/V voltage scaling slope and 88% at the 200μs/V voltage scaling slope. RazorProtector can help maintain Razor energy efficiency for a processor with a microsecond order voltage scaling restriction.

Thirdly, we furthered the study of adaptivity in RazorProtector by adding a tuning procedure to best fit the RazorProtector control to program characteristics. We evaluate EDP reduction for various applications from image processing programs and MiBench[26], under a practical voltage scaling speeds as 100μs/V and 40μs/V. Under a medium scaling speed 40uV/s, the adaptive method shows its efficiency by outperforming all static controls. In summary, 56% EDP reduc-

tion can be achieved by the enhanced RazorProtector method as compared to traditional DVS application, under high IR-drop zones.

Acknowledgements

The author would like to gratefully acknowledge Professor Yasuhiko Nakashima of Nara Institute of Science and Technology (NAIST) for supervising this dissertation, continuous support, suggestions and encouragement. The author would like to sincerely appreciate for his support, which gives him the opportunity to study in a doctoral course during the business in his company.

The author wishes to express his gratitude to Professor Michiko Inoue of NAIST for proof reading this dissertation. The author would like to sincerely appreciate for her valuable comments and critical comments.

The author would like to especially thank Assistant Professor Jun Yao of NAIST for his suggestions and continuous support, these have helped him very much throughout the production of this study. This dissertation owes much to his thoughtful and helpful comments.

The author also would like to express his appreciation to Assistant Professor Takashi Nakada of The University of Tokyo for his valuable opinions and support. This dissertation could not be accomplished without his understanding and support.

The author also would like to thank the members of NAIST Nakashima Laboratory for fruitful discussions and support.

Finally, the author would like to thank his wife, Hiromi Sasagawa from the bottom of his heart for continuous support and encouragement.

References

[1] International Technology Roadmap for Semiconductors, "2011 Overall Roadmap Technology Characteristics (ORTC)", ITRS 2011 Edition, http://www.itrs.net/Links/2011ITRS/Home2011.htm.

[2] Intel Corporation, "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor", white paper, http://download.intel.com/design/network/papers/30117401.pdf.

[3] T. Kehl, "Hardware Self-Tuning and Circuit Performance Monitoring", 1993 Int'l Conference on Computer Design (ICCD-93), pp. 188–192, October 1993.

[4] A. Uht, "Uniprocessor Performance Enhancement Through Adaptive Clock Frequency Control", 2003 International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, e-Medicine, and Mobile Technologies on the Internet (SSGRR 2003w), pp. 1–10, January 2003.

[5] G. Wolrich, E. McLellan, L. Harada, J. Montanaro, and R. Yodlowski, "A High Performance Floating Point Coprocessor", IEEE Journal of Solid-State Circuits, 19 (5), pp. 690–696, October 1984.

[6] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System", International Solid-State Circuits Conference, pp. 1571–1580, Feb. 2000.

[7] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active Management of Timing Guardband to Save Energy in POWER7", In Proceedings of the 44th International Symposium on Microarchitecture (MICRO44), pp. 1–11, Dec. 2011.

[8] T. Sato and Y. Kunitake, "A Simple Flip-Flop Circuit for Typical-Case Designs for DFM", 8th International Symposium on Quality Electronic Design, pp. 539–544, 2007.

[9] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "Razor: A low-power pipeline based on circuit-level timing speculation", In Proceedings of the 36th Annual IEEE/ACM Int. Symp. Microarchitecture, pp. 7–18, Dec. 2003.

[10] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance", IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 44, NO. 1, pp. 32–48, Jan. 2009.

[11] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, 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 Proceedings of the IEEE International Solid-State Circuits Conference, 15.6, pp. 284–286, Feb. 2010.

[12] T. Austin, "DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design", In Proceedings of the 32nd International Symposium on Microarchitecture (MICRO32), pp. 196–207, Nov. 1999.

[13] K. Sundaramoorthy, Z. Purser, E. Rotenburg, “Slipstream processors: improving both performance and fault tolerance”, In ASPLOS-IX Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, pp. 257–268, Nov. 2000.

[14] J. Xin and R. Joseph, “Identifying and Predicting Timing-Critical Instructions to Boost Timing Speculation”, In Proceedings of the 44th International Symposium on Microarchitecture (MICRO44), pp. 128–139, Dec. 2011.

[15] T. Sakurai, and R. A. Newton, “Alpha-Power Law MOSFET Model and its Application to CMOS Inverter Delay and Other Formulas”, IEEE J. Solid-State Circuits, VOL.25, No.2, pp. 584–594, April 1990.

[16] K. Inagaki, K. Kanda and T. Sakurai, “Organization of a standard SPICE model based on International Technology Roadmap for Semiconductors”, Proceedings of the Fundamentals Society Conference of IEICE, pp. 74–74, September 2000.

[17] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, “A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor”, In Proceedings of the International Symposium on Microarchitecture (MICRO), pp. 29–40, December 2003.

[18] K. R. Walcott, G. Humphreys, and S. Gurumurthi, “Dynamic Prediction of Architectural Vulnerability from Microarchitectural State”, In Proceedings of the International Symposium on Computer Architecture, pp. 516–527, June 2007.

[19] 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 31st Annual International Symposium on Computer Architecture (ISCA), pp. 264–275, June 2004.

[20] S. S. Mukherjee, C. Weaver, J. Emer, Steven K. Reinhardt, and T. Austin, “A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor”, In Proceedings of the 36th International Symposium on Microarchitecture (MICRO-36), pp. 29–40, Dec. 2003.

[21] K. Yoshimura, T. Iwakami, T. Nakada, J. Yao, H. Shimada and Y. Nakashima, “An Instruction Mapping Scheme for FU Array Accelerator”, IEICE Trans. on Information and Systems, Vol.E94–D, No.2, pp. 286–297, Feb. 2011.

[22] H. Li, C. Y. Cher, T. N. Vijaykumar and K. Roy, “VSV: L2-Miss-Driven Variable Supply-Voltage Scaling for Low Power”, In Proceedings of the 36th International Symposium on Microarchitecture (MICRO-36), pp. 19–28, Dec. 2003.

[23] W. Kim, D. M. Brooks and G. Wei, “A Fully-Integrated 3-Level DC/DC Converter for Nanosecond-Scale DVS with Fast Shunt Regulation”, In Proceedings of the IEEE International Solid-State Circuits Conference, pp. 268–270, Feb. 2011.

[24] S. Gurram, O. Brennan and T. Wilkerson, “DC-to-DC Switching-Regulator Insights-Achieving Longer Battery Life in DSP Systems”, Analog Dialogue Vol.41, No.4, pp. 11–15, Dec. 2007.

[25] Y. Sasagawa, J. Yao, T. Nakada and Y. Nakashima, “RazorProtector: Maintaining Razor DVS Efficiency in Large IR-drop Zones by an Adaptive Redundant Data-Path”, IEICE Trans. on Information and Systems, Vol.E95–A, No.12, pp. 2319–2329, Dec. 2012.

[26] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite”, Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on, pp. 3–14, Dec. 2001.

[27] J. Yao, S. Miwa, H. Shimada, and S. Tomita, “A Dynamic Control Mechanism for Pipeline Stage Unification by Identifying Program Phase”, IEICE Trans. on Information and Systems, Vol.E91–D, No.4, pp. 1010–1022, April 2008.

List of Publications

Journal Papers

1. Yukihiro SASAGAWA, Jun YAO, Takashi NAKADA and Yasuhiko NAKASHIMA, "RazorProtector: Maintaining Razor DVS Efficiency in Large IR-drop Zones by an Adaptive Redundant Data-Path", IEICE Trans. on Information and Systems, Vol.E95–A, No.12, pp. 2319–2329, Dec. 2012.

Conference and Workshops (Referred)

1. Yukihiro SASAGAWA, Jun YAO, Takashi NAKADA and Yasuhiko NAKASHIMA, "Improving DVS Efficiency by Tolerating IR-drops with an Adaptive Redundant Data-Path", In Proceeding of WRA 2011: 2nd Workshop on Resilient Architectures (in conjunction with MICRO-2011), pp. 1–8, Dec. 2011.

Conference and Workshops (Not Referred)

1. 笹川幸宏, 姚駿, 中田尚, 中島康彦, "演算器の適応的冗長化による高効率DVS方式の提案", In Proceeding of SWOPP 2011, 信学技報, vol.111, no.164, DC2011-15, pp. 1–6, Jul. 2011.