Understanding Variations for Better Adjusting Parallel Supplemental Redundant Executions to Tolerate Timing Faults

Yukihiro SASAGAWA^{†*}, Jun YAO^{†a)}, and Yasuhiko NAKASHIMA[†], Members

SUMMARY Razor Flip-Flop (FF) is a good combination for the dynamic voltage scaling (DVS) technique to achieve high energy efficiency. We previously proposed a RazorProtector scheme, which uses, under a very high IR-drop zone, a redundant data-path to provide a very fast recovery for a Razor-FF based processor. In this paper, we propose a dynamic method to adjust the redundancy level to fine-grained fit both the program behaviors and processor manufacturing variations so as to achieve an optimal power saving. We design an online turning method to adjust the redundancy level according to the most related parameters, ILP (Instruction Level Parallelism) and DCF (Delay Criticality Factor). Our simulation results show that under a workload suite with different behaviors, the adaptive redundancy can achieve better Energy Delay Product (EDP) reduction than any static controls. Compared to the traditional application of Razor-FF and DVS, our proposed dynamic control achieves an EDP reduction of 56% in average for the workloads we studied.

key words: adaptive redundancy, setup error recovery, DVS, low power, AVF, ILP

1. Introduction

To alleviate the power problems that come along with the aggressive increase of semiconductor integrity, dynamic voltage scaling (DVS) [1]–[4] technologies have now become default-assembled in commercial chips and activated aggressively. Also, Razor Flip-Flop (FF) [5]–[8] has gained popularity to cope with the DVS technique to further reduce sufficiently the voltage by using a set of primary/shadow latches in the critical computation paths. Razor-FF can help detect setup errors so as to avoid unnecessary guard-bands. A maximum reduction of power consumption can be achieved when the cost of setup-error recovery and the gaining of voltage down-scaling reach a balancing point, when most voltage design-margins have been aggressively taken out. However, this energy-effective near-0-margin working mode of Razor-FF with the DVS comes at a cost of increased vulnerability to sudden and unexpected IR-drops. When frequent IR-drops occur, the processor is required to re-tune its balancing point, as the previously balanced voltage is not sufficient to generate correct results. The voltage re-tune, with several voltage changes, requires a long delay, which will be in the order of microseconds in modern processors. The performance will downgrade largely before the new bal-

ancing point is reached.

Previously, we proposed RazorProtector [9] that uses an architectural redundant-path as an alternative method to address this long timing-fault re-balancing delay. The redundant path is shown in Fig. 1, where the originally three-issue processor with three pipelines are configured into one P-PIPE and two R-PIPEs (Fig. 1(a)). The R-PIPEs work on the same instruction stream as the normal path P-PIPE, while the frequency in R-PIPEs is halved to guarantee a setup-error-free execution in the R-PIPEs, as shown in Fig. 1(b). More specifically, under the detection of a timing fault due to the unexpected IR-drops, data in the redundant R-PIPE can be used to fast recover from the erroneous execution, so that a voltage up-scaling in the traditional Razor-FF processor is not necessary. By not having the voltage up-scaling, RazorProtector can avoid a sharp performance degradation, especially under frequent IR-drops.

In this paper, based on the original RazorProtector, we try a dynamic tuning method of controlling the application

(a) Structure of Redundant data-path in RazorProtector

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

Fig. 1 RazorProtector: redundancy data-path for a fast timing-fault recovery.

Manuscript received January 11, 2014.

Manuscript revised April 28, 2014.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.

*Currently, with Panasonic Corp.

a) E-mail: yaojun@is.naist.jp (Corresponding author)

DOI: 10.1587/transinf.2014PAP0024

of parallel and redundant modes in RazorProtector. The control is based on the sampling data of the current setup-timing error rate, the program operation vulnerability to timing errors, and the performance impact from the redundant execution. More specifically, the most sensitive parameters to the final power efficiency, the ILP (Instruction Level Parallelism), the DCF (Delay Criticality Factor), which is a critical path related metric, and the program stable hotspot duration, are tightly reflected into the dynamic control algorithm. The execution variations caused by the distribution of instructions with different critical paths, the difference in the various input data of the long-delay operations, and the processor manufacturing variations, are expected to be finally covered in the dynamic tuning for a better control of the parallel and redundant modes. 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.

The paper is constructed as follows. Section 2 introduces the background technique, RazorProtector, and our timing-fault risk level model in common microprocessors. Section 3 presents the redundancy-adapting algorithm by using dynamic ILP and DCF studies. Section 4 presents the energy and performance study of the RazorProtector, compared with the conventional DVS application and static RazorProtector methods. Section 5 concludes the paper.

2. RazorProtector: Using Redundancy Path to Fast Recover the Timing Faults

2.1 Architecture of RazorProtector

This section reviews the RazorProtector framework. The whole architecture is illustrated in Fig. 1, which represents a three-issue processor. The IF and ID in Fig. 1 are able to fetch and decode three instructions per cycle. The three execution modules are ready to process instructions that are clear of dependence constraints, exploiting instruction level parallelism (ILP) for a maximum 3x the performance of the single-issue processor. However, due to the unexpected IR-drops, all of these three execution pipelines are vulnerable to the timing faults. Though we can add Razor-FF to detect the timing faults, a time-consuming recovery, as shown in Fig. 2, is required to tolerate the timing faults. When frequent IR-drops happen near the critical voltage balanced zone, an even more time-consuming voltage re-scaling is required to put the processor in a healthy working mode.

The RazorProtector, based on redundant data-path, has been proposed to solve the timing-consuming recovery problem, as introduced in paper [9]. The redundant data-path structure is also illustrated in Fig. 1(a). More specifically, the original three execution pipelines will be divided into two categories: 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. The R-PIPEs are designed to use two cycles to finish each calcu-

Fig. 2 Recovery scenario in traditional Razor-FF processors, under a timing fault.

Fig. 3 Redundant parallel data-paths, used when IR-drops are high and critical. Maximum IPC is 1 but can fast recover from timing faults.

lation 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. Figure 1(b) shows the redundant instruction-dispatch in the three pipelines. An instruction will be dispatched into the P-PIPE and also into one of the R-PIPEs, according to the occupancy of R-PIPEs. Because R-PIPE processes one instruction per two cycles, in the even cycles the duplicated instructions go to the first R-PIPE and in the odd cycles go to the second R-PIPE, respectively.

Now the redundant data-path in the RazorProtector dispatches one instruction in the P-PIPE each cycle and the same instruction into the two 50% slower R-PIPEs in the two consecutive cycles. Although the total throughput of this redundant mode within the three pipelines is $IPC = 1$, the redundant path can help provide a very fast recovery, as shown in Fig. 1. More specifically, under a timing fault, the result calculated in timing-fault vulnerable P-PIPE will be discarded. Instead, later instructions in the P-PIPE will refer the results in the R-PIPE that works on the same instruction. Only a one-cycle bubble is needed to wait for the completion of the R-PIPE processing. This fast recovery makes the RazorProtector a promising alternative to avoid

sharp performance degradation under frequent IR-drops.

Overall, a RazorProtector processor work under two modes, the parallel mode (P_mode) and the redundant mode (R_mode). The P_mode uses the three-issue in maximum for a parallel execution, while the R_mode uses the redundant execution for a fast timing-fault recovery. The data-path, also known as the application binary, does not need to change between the two modes. Instead, under the R_mode, a hardware module in the issue-logic is used to dispatch an instruction into the P-PIPE and to also dispatch a duplicated instruction into one of the two R-PIPEs. The forwarding logic is also changed a little to additionally select the correct R-PIPE result for the later instructions if a timing fault occurs in the P-PIPE.

2.2 Modeling the RISK Level

This section reviews key metrics, such as Delay Criticality Factor (DCF), in the RazorProtector [9] paper, to help model the timing-fault vulnerability level in this work. Specifically, DCF, which is a measure of operating units to timing faults, and the related RISK level to timing faults are calculated as follows:

$$\Pr_{\text{path}} = \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{t_{\text{delay}} - t_{\text{typdelay}}}{\sqrt{2}\sigma^2} \right) \right) \quad (1)$$

$$\text{DCF} = \frac{\sum (\Pr_{\text{path}} \times N_{\text{path}})}{\#B} \quad (2)$$

$$\text{RISK}_{\text{setuperr}} = \text{ERR}_{\text{setup}} \times \text{DCF} \quad (3)$$

(Notes: erf is error function)

Figure 4 shows the relationship between logic structure and above equations. Operation unit A is correspondent to a certain DCF defined by Eq. 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, $\Pr_{\text{path}}$ is the probability of setup errors, which are caused by the delay of each critical path t_{delay} . In each operation unit (e.g. ALU in data-path), the $\Pr_{\text{path}}$ of the activated path can be used to indicate its current setup-error possibility. Equation 1 assumes that the delays from all critical paths follow a normal distribution around the average delay t_{typdelay} . According to Eq. 1, $\Pr_{\text{path}}$ will be 100% in the case that t_{delay} is the same as the maximum delay in all critical paths. In Eq. 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 derived by combining the $\Pr_{\text{path}}$ of each path. Each operation unit will have a corresponding DCF, which measures its vulnerability to the setup error.

Table 1 presents an example of DCF for each instruction. This table also contains operation delay, which is average of t_{delay} corresponding to instruction. Each instruction has the DCF value respectively, which has a correlation with

Fig. 4 Relationship between logic structure and equation.

Table 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.)

operation delay described as $\Pr_{\text{path}}$. However, some instructions (e.g. ADD and ASR) have different correlations. This is caused by the difference of the logic structure, described as N_{path} and $\#B$ in Eq. 2. In this paper, the variation to the calculation delay from the input values has not been considered. The statistical DCF in this paper 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 the same for either a static or a dynamic DCF.

In Eq. 3, $\text{ERR}_{\text{setup}}$ is the setup error rate observed in the system (e.g. by error detection flip-flops). The risk of a 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 $\text{ERR}_{\text{setup}}$.

Since DCF determines the vulnerability of each operation to the setup error, it is possible to use its value to switch adaptively between parallel and redundant modes (P_mode and R_mode) according to the detailed instruction flow. A DCF threshold is used to define the boundary in order to switch the mode, as in Eq. 4.

$$\text{DCF}_{\text{th}} = \frac{\text{RISK}_{\text{th}}}{\text{ERR}_{\text{setup}}} \quad (4)$$

3. Our Proposal: A Dynamic RISK Tuning Based Redundancy Adjusting

3.1 Program Parameters to Select P_mode or R_mode in RazorProtector

The Parallel mode (P_mode) in RazorProtector uses the

Fig. 5 Increase of energy consumption due to IR-drop.

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 Fig. 5 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 the voltage change period is calculated by following the alpha-power-law delay model [10], [11]. The y -axes show the normalized energy consumption.

It can be easily observed from Fig. 5 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 the setup-error recovery is frequently required.

The difficult turning-point is at the low error-rate part. As given in Fig. 5, 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 cover the performance loss in the R_mode, in which the ILP is not exploited due to redundancy.

We then try a quantitative study of these trade-off considerations between the P_mode and the R_mode. More specifically, when considering the energy-delay-product (EDP) as a measure, the P_mode and the R_mode will have the following balancing-point under a given error rate, as:

$$\frac{EDP(P_{mode})}{EDP(R_{mode})} = \frac{N_I / IPC_{P_{mode}} + \#_{of_errors} \times n_{depth}}{N_I / 1 + \#_{of_errors} \times 1} \quad (5)$$

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, when the large DCF of instructions makes data arrive later than the setup requirement. $\#_{of_errors}$ actually reflects the average DCF of this workload. $IPC_{P_{mode}}$ 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

Fig. 6 Relationship of the loop duration and the required interval for updating $RISK_{th}$.

the working mode of R_mode, the $IPC_{P_{mode}}$ has a constant value of 1. However, the recovery penalty under R_mode is also 1 cycle, which is lower than the n_{depth} penalty under P_mode.

Under some extreme cases where $\#_{of_errors} = 0$, the R_mode has similar energy efficiency as P_mode only when $IPC_{P_{mode}} = 1$, which indicates a very low ILP workload. When $n_{depth} = 5$ and 12.5% of the instructions are with very large DCF to cause faults, P_mode should have an $IPC_{P_{mode}} > 2$ to be better than R_mode. These very rough calculations also mirror our method of using performance counters—the ILP and the number of visible errors—to give an estimation of the successive working mode.

3.2 Algorithm to Tune the Adaptive Redundancy

In this section, we are using architectural method to give a dynamic redundancy control of the RazorProtector. Figure 6 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, which give recursive program characteristics. Both ILP and DCF will be stable for a long time and will then shift to other values after a sudden change. Accordingly, Fig. 7 shows our enhanced control architecture.

“A” in Fig. 7 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 [9]. The DCF of this pending instruction will be compared to the threshold DCF_{th} to determine the suitable redundancy level, as P_mode and R_mode. As introduced in Sect. 3.1, DCF_{th} should be tuned to fit for the program characteristics and the error rate to achieve optimal energy efficiency by using the RazorProtector. In the

Fig. 7 Determine redundancy by DCF, ERR_{setup} and RISK_{th}.

architecture shown in Fig. 7, we use an error-rate sampler to gather the error detection signals generated from error detecting flip-flops, as (B) in Fig. 7. The value of ERR_{setup} is then calculated according to the number of collected errors in the sampling period. A suitable RISK_{th} is predicted accordingly, and DCF_{th} can then be easily given by following Eq. 4.

Figure 8 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. 8, at each decoding stage (Fig. 7 (A)), the decoder will give the DCF of the current instruction group, and the 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.

The signal “berror” in Fig. 8 is then used to indicate whether there is a 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 its loss of the instruction-issue chance is $IPC \times n_{depth}$, with weighted factor. The value n_{depth} is the pipeline depth, which is related 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” increment is 1 (Fig. 8). However, when there is no “berror” in the P_{PIPE} execution, the loss

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

while (1) { /* cycle */
    inst[] = fetch_inst();
    /* Fig. 7 (A) */
    curr_mode = (DCF > DCFth) ? R_mode : P_mode;

    berror = exec(inst[], curr_mode); /* Fig. 7 (B) */
    /* Fig. 7 (2) */
    if (berror) {
        recover(inst[], curr_mode);
        xscore[P_mode] += ndepth × 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. 7 (3) */
        if (xscore[P_mode] > xscore[R_mode])
            RISKth -= STEP;
        else
            RISKth += STEP;
        n = 0;
        xscore[P_mode] = xscore[R_mode] = 0;

        DCFth = RISKth / ERRsetup; /* Fig. 7 (4) */
    } /* end of reconfiguration */
} /* end of this cycle */
```

Fig. 8 Tuning algorithm to find optimized DCF_{th} and setting redundancy mode.

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. Generally, under the same ERR_{setup}, a smaller RISK_{th} will result in a smaller DCF, and gives accordingly 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 paper, RISK_{th} varies from 0.01% to 0.1% and then to 1% respectively. Accordingly, the STEP is respectively 0.01% when RISK_{th} being within 0.01% to 0.1, and is 0.1% when RISK_{th} being within 0.1% to 1%.

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 can be selected more, with a help of a slowly incrementing “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. 8. A higher average DCF will have a larger chance to satisfy the “berror == true” condition and P_mode will get more penalty, both finally resulting into a decreasing tendency in DCF_{th} . Therefore, according to these scenarios, we have included all possible tuning parameters, as ILP, DCF, and ERR_{setup} by the help of this on-line tuning algorithm. Note that this algorithm contains only simple strategies with a limited number of historical statistical parameters. The implementation of this algorithm is therefore not high cost.

4. 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 [9]–[11]. The parameters of our simulator are listed in Table 2. Note that the baseline processor is designed to have three execution pipelines, which work on ALU and SHIFT operations, and a load/store pipeline. Due to the setting of the address space, the address calculation unit in the load/store pipeline has a narrower width than the three execution pipelines. We thus regard that only the arithmetic and media units in the three execution pipelines will have timing faults due to the long critical paths. RazorProtector are used in those three pipelines to aid a fast recovery for the timing faults.

The corresponding circuit data is extracted from a special FR-V processor [12]. 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 if there is not error.
2. The voltage keeps the current level when the error rate is under a tolerable value.
3. The voltage up-scales when the error rate is intolerably high. The up-scaling voltage step corresponds to “Voltage scaling slope” in Fig. 9.

4.1 Effectiveness of the $RISK_{th}$ 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 bal-

Table 2 Parameters of cycle-based simulator.

Instruction issue width	4/cycle (3 ALUs/SHIFTS, 1 LD/ST)
No. of General purpose register (GR)	32
No. of Media register (MR)	32
L2 $\Rightarrow$ L1 transferring	8bytes/cycle
Instruction cache	4way 16KB (64bytes/line)
L1 cache	4way 16KB (64bytes/line)
L1 $\Rightarrow$ L0 transferring	12bytes/cycle
Store buffer	4entry

Fig. 9 EDP results of different $RISK_{th}$ (unsharp, 10% IR-drop).

anced 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.

We give the EDP result of the benchmark *unsharp* in Fig. 9, as the representative workload to illustrate the results of non-RazorProtector, static and enhanced RazorProtectors. 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. 9. The voltage scaling speed is given in the horizontal axis, which is the time duration used in each unit-voltage scaling. Note that the redundancy mode is used only before the voltage balancing point is re-adapted. After that, the normal multi-issue parallel execution will take position under the balanced voltage V_{opt} .

Basically, for *unsharp*, $RISK_{th}$ 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 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 $RISK_{th}$.

In Fig. 9, the EDP results of the dynamically adapted $RISK_{th}$ by the tuning method given in Sect. 3.2 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 our tuning method, the program execution has been properly controlled, 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.

4.2 EDP Reduction Results

This section gives a summary of the EDP reduction that the adaptive RazorProtector can achieve, by applying it onto the benchmarks we studied. In this paper, the traditional Razor-FF [6] without any pipeline redundancy serves as the baseline processor. This paper then compares with a previous research RazorProtector in [9], which uses a fixed $RISK_{th}$. Both EDPs of the previous static RazorProtector and this dynamic one are normalized by the traditional unprotected Razor-FF 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 Sect. 4.1. *FI-a* and *FI-b* are parts of a frame interpolation, where *FI-a* is searching block corresponding to minimum SAD, which contains mainly comparison instructions. *FI-b* is interpolating pixels corresponding to the 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 [13]. These programs cover the Automotive and Industrial Control category (*basicmath*, *qsort*, *susan*), the Network category (*patricia*), Security (*sha*) and the Consumer devices category (*jpeg*). All these benchmarks make up the various workloads to study the final EDP reduction results of the $RISK_{th}$ adapting RazorProtector approach.

Fig. 10 EDP reduction in various applications.

The EDP results of these benchmarks are shown in Fig. 10, 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 $100\mu s/V$ 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 the adaptive RazorProtector can be used to maintain the applicability of DVS even when the unexpected IR-drop reaches a relatively high level.

It can also be observed from Fig. 10 that the ability of furthering EDP reduction by our dynamic tuning in RazorProtector is largely varying in workloads. The benchmarks are listed in Fig. 10, following a decreasing EDP order. Among all the benchmarks, the tuned adaptive RazorProtector gets 17% reduction in *unsharp*, but it achieves near 100% reduction in *basicmath* and *patricia*. As has been introduced in Sect. 3.1, the applicability of R_mode in RazorProtector connects to the program characteristics, ILP and DCF. Here, we try to use the study of program characteristics, as shown in the up-right subfigure Fig. 10 (a), to present a rough analysis of ILP and DCF of all the benchmarks. The horizontal direction of Fig. 10(a) gives the IPC difference. Benchmarks in Group A have larger IPCs than those in Group B. The vertical direction in the subfigure Fig. 10(a) 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 the DCFs in the workload vary 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). From these data, we can say that our adaptation algorithm in Sect. 3.2 correctly recognizes the changes of the most suitable program characteristics to the setup errors, and thus it can adapt optimally the redundancy level of RazorProtector for good energy efficiency.

The DCF values of the group B in Fig.10(a) are the averaged DCF data from each individual program execution. In addition, we find in Fig.10 that all group B benchmarks, which are from the MiBench programs, have better EDP reductions after applying the adaptive RazorProtector than the filter programs. Especially, the application of RazorProtector in *basicmath* and *patricia* achieve an EDP reduction near 100%. From another view point, it implies that the performance of the non-redundant execution with traditional DVS and Razor-FF has suffered largely in *basicmath* and *patricia* under a high setup-error rate. More specifically, according to our experiment settings, the voltage will always try to aggressively go down when executing short critical-path instructions and go up when setup errors frequently occur. The original unprotected Razor-FF and DVS in programs *basicmath* and *patricia* have experienced a lot of unnecessary voltage up-scaling and down-scaling due to the fine-grained voltage control. To explain this, we list the DCFs of most time-consuming loop bodies in Table 3. Note that the high DCFs will tend to make the execution stay at the low-voltage zone, and vice versa. But when DCFs are at the boundary zone, the skewing effect of frequent voltage-change may occur. From the data in Table 3, we can tell that the DCF values ranged from 17.9 to 18.7 are likely to result in very frequent voltage-scaling.

Though we can change the voltage scaling policy in the original Razor-FF and DVS to make it less fine-grained, the frequent voltage change may always possibly occur at some other boundary threshold. However, the concept of RazorProtector is using the redundant path instead of the voltage change to tolerate timing faults. It tends to reduce the unnecessary voltage scaling. In addition, our online-tuning method can further adaptively control the voltage change in the RazorProtector according to a combined measurement, as “xscore” in Fig. 8. From the view point of eliminating adverse voltage skews, the proposed online tuning method is necessary.

Figure 11 gives further comparison between the static and the adaptive RazorProtector applications, under a voltage scaling speed of $100\mu\text{V}/\text{V}$. Due to the relatively slow voltage scaling speed, low RISK_{th} 0.01% and 0.1% are more 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. 10. Among all, our 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 to get a clear difference between P_mode and R_mode. The slow $100\mu\text{V}/\text{V}$ voltage changing speed ac-

Table 3 DCF values of top hot-loops in MiBench.

Bench.	Top two hot-loops	
	DCFs	time consumed
jpeg	20.0	44%
	18.1	20%
sha	20.9	32%
	17.9	32%
susan	20.9	32%
	19.8	26%
basicmath	18.7	47%
	17.9	5.0%
patricia	17.9	59%
	18.7	19%

Fig. 11 EDP comparison between fixed and adaptive RISK_{th} ($100\mu\text{s}/\text{V}$ voltage scaling slope).

Fig. 12 EDP comparison between fixed and adaptive RISK_{th} ($40\mu\text{s}/\text{V}$ voltage scaling slope).

tually requires a more preference of R_mode, which may thus cause a difference in the real application and algorithm determination. 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 $\text{EDP}_{\text{RISKth}=0.01\%} / \text{EDP}_{\text{RISKth}=adaptive}$ in reducing the EDP.

Figure 12 shows EDP comparison under a practical voltage scaling speed as $40\mu\text{s}/\text{V}$. 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 giving 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%, after normalized by the traditional DVS application.

5. Conclusions

This paper proposed the online turning algorithm for the adaptive 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 the adaptive 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, under a practical voltage scaling speeds as $100\mu\text{s}/\text{V}$ and $40\mu\text{s}/\text{V}$. Under a medium scaling speed $40\mu\text{s}/\text{V}$, 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.

Acknowledgments

This work is supported by VLSI Design and Education Center (VDEC), University of Tokyo with the collaboration of Synopsys Corporation, Cadence Design Systems, and Mentor Graphics. This work is supported by KAKENHI (No. 24240005, No. 24650020, and No. 2370060), and the STARC IS program.

References

- [1] T. Kehl, "Hardware self-tuning and circuit performance monitoring," 1993 Int'l Conference on Computer Design (ICCD-93), pp.188–192, Oct. 1993.
- [2] 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, Jan. 2003.
- [3] G. Wolrich, E. McLellan, L. Harada, J. Montanaro, and R. Yodlowski, "A high performance floating point coprocessor," IEEE J. Solid-State Circuits, vol.19, no.5, pp.690–696, Oct. 1984.
- [4] 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," Proc. 44th International Symposium on Microarchitecture (MICRO44), pp.1–11, Dec. 2011.
- [5] 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.
- [6] 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," Proc. 36th Annual IEEE/ACM Int. Symp. Microarchitecture, pp.7–18, Dec. 2003.
- [7] 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 J. Solid-State Circuits, vol.44, no.1, pp.32–48, Jan. 2009.
- [8] D. Bull, S. Das, K. Shivshankar, G. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," Proc. IEEE International Solid-State Circuits Conference, 15.6, pp.284–286, Feb. 2010.
- [9] 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. Fundamentals, vol.E95-A, no.12, pp.2319–2329, Dec. 2012.
- [10] 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.
- [11] K. Inagaki, K. Kanda, and T. Sakurai, "Organization of a standard SPICE model based on international technology roadmap for semiconductors," Proc. Fundamentals Conf. IEICE, A-3-7, Sept. 2000.
- [12] K. Yoshimura, T. Iwakami, T. Nakada, J. Yao, H. Shimada, and Y. Nakashima, "An instruction mapping scheme for FU array accelerator," IEICE Trans. Inf. & Syst., vol.E94-D, no.2, pp.286–297, Feb. 2011.
- [13] 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," 2001 IEEE International Workshop on, Workload Characterization, 2001. WWC-4, pp.3–14, Dec. 2001.

Yukihiro Sasagawa received the B.E., M.E. degrees from Doshisha University in 1992 and 1994 respectively. He is a signal processor architect in the Semiconductor Department, Panasonic Corporation since 1994. He received the D.E. degree from Graduate School of Information Science, Nara Institute of Science and Technology in 2013. His research interests include power-efficient signal processing, CMOS circuit design, and processor architecture.

Jun Yao received the B.E., M.E. degrees from Tsinghua University in 2001 and 2004 respectively. He received the Ph.D. degree from Kyoto University in 2009. Since 2013, he has been an associate professor in Graduate School of Information Science, Nara Institute of Science and Technology. His current researches are in computer architecture field, especially low-power consumption and high-reliable architectures. He is a member of IEEE, ACM and IPSJ.

Yasuhiko Nakashima received the B.E., M.E. and Ph.D. degrees in Computer Engineering from Kyoto University, Japan in 1986, 1988 and 1998 respectively. He was a computer architect in the Computer and System Architecture Department, FUJITSU Limited in 1988–1999. From 1999 to 2005, he was an associate professor in the Graduate School of Economics, Kyoto University. Since 2006, he has been a professor in the Graduate School of Information Science, Nara Institute of Science and Technology. His

research interests include processor architecture, emulation, CMOS circuit design, and evolutionary computation. He is a member of IEEE CS, ACM and IPSJ.