Mitigating Performance Loss in Aggressive DVS Using Dual-Sensing Flip-Flops

Kunitake, Yuji
Kyushu University

Sato, Toshinori
Fukuoka University | Kyushu University

Yasuura, Hiroto
Japan Science and Technology Agency, CREST | Kyushu University

出版情報：VLSI-SoC. 2008, pp.543-546, 2008-10-15
バージョン：
権利関係：

Mitigating Performance Loss in Aggressive DVS Using Dual-Sensing Flip-Flops

Yuji Kunitake [†] Toshinori Sato [‡ † §] Hiroto Yasuura [† §]
[†] Kyushu University [‡] Fukuoka University
[§] Japan Science and Technology Agency, CREST
y-kunitake@c.csce.kyushu-u.ac.jp

***Abstract*— Traditional worst-case design is becoming much difficult. The increase in parameter variations requires large design margins. However since the worst-case situations do not always occur, the design margin is often overestimated. To eliminate the excessive design margin, designers should focus on typical case rather than worst case. We are investigating canary logic that is one of the typical-case designs. The canary logic can eliminate the excessive design margin by combined with Dynamic Voltage Scaling (DVS) system. A naive combination of the canary and the DVS suffers serious performance loss due to useless voltage scaling. In this paper, we show where the performance loss comes from and improve the canary logic to eliminate the excessive design margin without performance loss.**

I. INTRODUCTION

The deep submicron semiconductor technologies increase parameter variations and thus the processer design becomes more difficult [1,5,8,9]. Generally, processor's maximum clock frequency is determined by considering the worst-case critical path delay and a safety margin. The margin is required since delays are not constant due to parameter variations. The parameter variations include process, voltage, and temperature (PVT) variations. In the worst-case design, the critical path delay and the margins are summed up and thus PVT variations have a serious impact on supply voltage to satisfy required operating frequency and to improve timing yield of microprocessors. In other words, managing parameter variations is a key to power reduction.

In the worst-case design, the parameters rarely become the worst case. Considering the situations, we are investigating typical-case design methodologies, which consider typical cases rather than worst cases. In the typical-case design, the margin is estimated by the typical case of parameters and thus the constraints in processer design are relaxed. If the worst case occurred, the processor would cause timing errors. Thus, we have to assure the correct processor state by avoiding or by recovering from timing errors.

Razor [2] is one of the typical-case designs, which permits to violate timing constraints to improve energy efficiency. Razor works at higher clock frequency than that determined by the critical path delay. In order to detect timing errors, Razor flip-flop (FF) is proposed. Each timing-critical FF (main FF) has its shadow FF, which is expected to always hold correct values. If the values latched in the main and shadow FFs do not match, a timing error is detected. When the timing error is detected in microprocessor pipelines, the processor state is recovered to a safe point.

However, Razor can be further improved. We are investigating canary logic [7], which does not detect but predict timing errors by using canary FFs. Thus, the canary logic does not need any recovery processes. We will find later in the present paper that adopting the canary logic and the DVS on a carry select adder (CSLA) shows the potential in energy reduction of 31%. Unfortunately, the current canary logic causes severe performance loss. In this paper, we analyze where the performance loss comes from and solve the performance loss problem.

This paper is organized as follows. Section II introduces the canary logic. Section III explains the evaluation environment. Section IV describes a problem in the canary logic. Section V proposes dual-sensing canary to solve the problem. Section VI presents experimental results. Finally, Section VII concludes this paper.

II. CANARY LOGIC

The increase in parameter variations requires large design margins. In order to eliminate the excessive design margin, designers should focus on typical case rather than worst case. As a typical-case design, we are investigating canary FF [7].

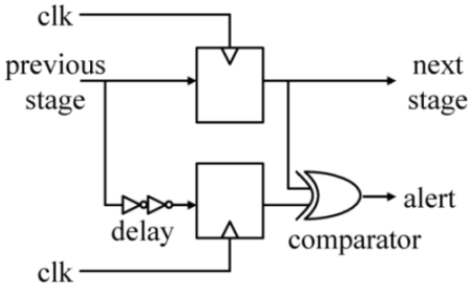

Figure 1. Canary Flip-Flop

The canary FF is augmented with a delay element and a shadow FF, as shown in Figure 1. The canary FF is used to detect a timing error. Every timing error is predicted by comparing the main FF value with that of the shadow FF, which runs into the timing error a little bit before the main FF.

Figure 2 explains how the DVS technique utilizes the canary FFs. The horizontal and vertical lines present time and supply voltage, respectively. At regular intervals, the supply voltage is decreased if a timing error is not predicted during the last interval. When a timing error is predicted to occur, the supply voltage is increased.

Figure 2. Canary's DVS

III. EVALUATION ENVIRONMENT

As a case study of the canary logic, we apply the canary logic to a CSLA. In order to evaluate the DVS system using canary logic, accurate circuit delay has to be considered. On system designs, architectural-level simulators are a must-be tool, which makes designers examine a wide variety of design choices. Unfortunately, current architectural-level simulators do not consider circuit delay and are inappropriate to evaluate the DVS system using canary logic. In contrast, gate-level simulation can estimate accurate circuit delay. However, the gate-level simulations require huge amount of simulation time. Therefore, we built a co-simulation environment, which combine gate- and architectural-level simulators.

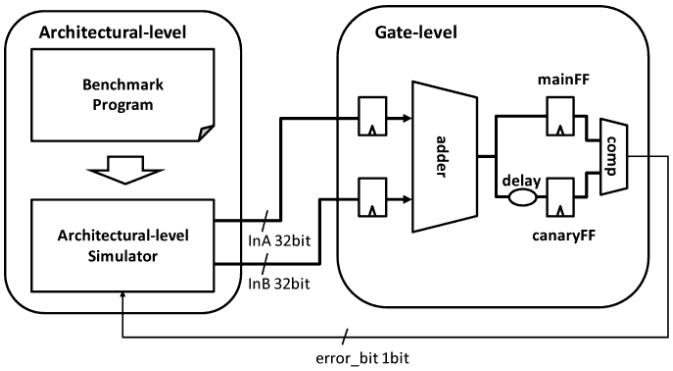

Figure 3. Co-simulation Framework

Figure 3 shows the framework of our co-simulator. The left block is an architectural-level simulator, where the behavior of the entire processer is simulated. The right block is a gate-level simulation environment, where the CSLA with the canary logic is simulated. The co-simulator behaves as follow.

1. When an ADD or a SUB instruction is executed in the architectural-level simulator, the gate-level simulator is triggered with its operands.
2. The CSLAs in the gate-level simulation execute the operation with the operands and the canary FFs examines timing errors.
3. Gate-level simulation results are delivered to the architectural-level simulator.
4. The supply voltage changes according to the gate-level simulation results.

The co-simulation is executed by repeating these steps from 1 to 4. In this way, when the delay information is necessary, the architectural-level simulator asks the gate-level simulator whether any timing errors are predicted or not.

TABLE I. PROCESSOR CONFIGURATIONS

Clock frequency	2 GHz
Fetch width	8 instructions
L1 instruction cache	16K, 2way, 1 cycle
Branch predictor	bimodal
Bimodal predictor	4K entries
Branch target buffer	1K sets, 4way
Dispatch width	4 instructions
Instruction window size	128 entries
Issue width	4 instructions
Integer ALUs	4 units
Integer multiplires	2 units
Floating ALU	1 unit
L1 data cache ports	2 ports
L1 data cache	16K, 4way, 2 cycles
Unified L2 cache	8M, 8way, 10 cycles
Memory	Infinite, 100 cycles
Commit width	8 instructions

TABLE II. FREQUENCY – VOLTAGE SPECIFICATIONS

F(GHz)	2.1	1.8	1.6	1.4
V_{dd}(V)	1.340	1.276	1.228	1.180
F(GHz)	1.2	1.0	0.8	0.6
V_{dd}(V)	1.132	1.084	1.036	0.988

The MASE simulator [6], is used for architectural-level simulation. TABLE I summarizes processor configurations. Seven integer programs from SPEC2000 CINT benchmark are used. For each program, 1 billion instructions are skipped before an actual simulation begins. After that each program is executed for 1 billion instructions.

In the gate-level simulation, we design a 32bit CSLA. SYNOPSYS DesignCompiler logic-synthesizes the CSLA with Hitachi 0.18μm standard cell libraries. The combinations of the clock frequency and the supply voltage of Intel Pentium M [3], which is shown in TABLE II, are used. We project the highest clock frequency, which is determined by CSLA's critical path delay reported by DesignCompiler and safety margin for PVT variations, onto Pentium's highest clock frequency. The safety margins of 50% and 100% of the CSLA's critical path delay are assumed. In other words, we assume that clock cycle time is 1.5 time and 2 time of the CSLA's critical path delay. The original canary's delay is 10% of the critical path delay. *CanaryS* and *canaryL* have 5% and 15% of the critical path delay as their delay, respectively. We use the interval of 100K clock cycles. It is assumed every supply voltage switching requires 10μs [4].

IV. Problem in Canary Logic

In this section, we show the potential in energy reduction and a problem in the canary logic.

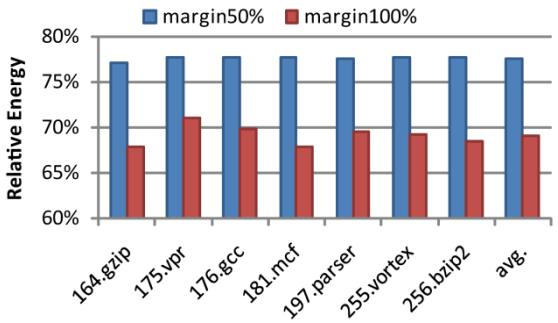

Figure 4. Energy Consumptions

Figure 4 shows energy consumptions. Each bar represents energy consumption relative to that of the baseline model. *margin50%* and *margin100%* mean the safety margin of 50% and 100% is provided. It is relative to the critical path delay. In other words, we assume that clock cycle time is 1.5 time and 2 time of critical path delay. In *margin50%*, energy can be reduced by 23% on average. In *margin100%*, energy can be reduced by 31% on average.

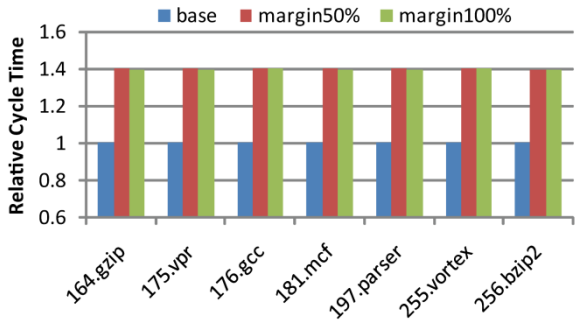

Figure 5. Execution Cycles

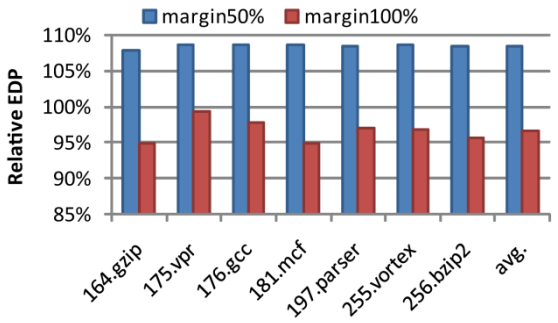

Figure 6. Energy Delay Product

In contrast, execution cycles are increased by about 40% for all programs, as shown in Figure 5. Each graph is normalized by the baseline execution cycles. Consequently, energy efficiency is diminished. Figure 6 shows Energy Delay Product (EDP). Each bar is normalized by the baseline EDP. In *margin50%*, EDP is increased by about 8% for all programs. The cause is performance overhead due to voltage scaling. Every change in the voltage requires 10μs. The voltage scaling strategy in the present paper shows an oscillation in supply voltage. Since every supply voltage switching makes processor unavailable during the transition, this oscillation has a serious impact on performance and hence on power efficiency. In order to avoid the supply voltage oscillation, the canary's DVS policy should be improved.

V. Dual-Sensing Canary

In order to avoid the performance loss due to the supply voltage oscillation, we improve the canary logic. We call the improved one dual-sensing canary. We change the voltage scaling strategy.

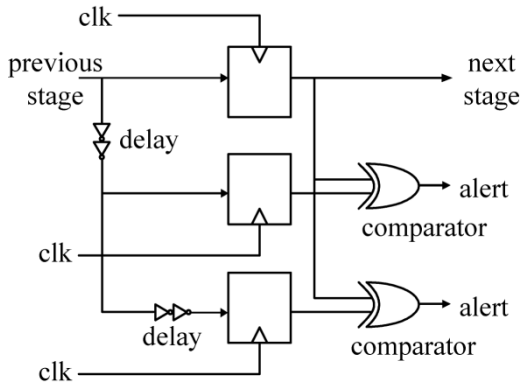

Figure 7. Dual-Sensing Canary

Figure 7 shows the dual-sensing canary. It has two shadow FFs, which are different in delay values. One has a smaller delay and is called *canaryS*. The other has a larger delay and is called *canaryL*. Hence, *canaryL* meets a timing error before *canaryS* does. In addition, we change the voltage scaling policy, as shown in TABLE III. If *canaryL* predicts an error and *canaryS* does not predict it, the supply voltage is unchanged in the next interval. If both *canaryL* and *canaryS* predict errors, it implies that there is small margin in the main FF. Thus, in order to avoid the error in main FF, the supply voltage is increased. When any errors are not predicted during the current interval in both canary FFs, there is sufficient timing margin and hence the voltage is decreased.

TABLE III. Voltage Scaling Policy

CanaryS	CanaryL	Supply voltage
Not predict	Not predict	Decreased
Not predict	Predict	Unchanged
Predict	Predict	Increased
Predict	Not predict	—

An example of the voltage scaling using the dual-sensing canary is shown in Figure 8. Horizontal line shows time line. Vertical line shows supply voltage. Each Ix shows interval.

When there is an excessive timing margin, the supply voltage is decreased. At t1, only *canaryL* predicts a timing error. Thus, in the next interval I3, the supply voltage is unchanged. At t2, *canaryS* predicts a timing error. Thus, the supply voltage is increased. In the next interval I4, the supply voltage is unchanged, because only *canaryL* predicts a timing error at t3. As you can see, we expect that useless voltage scaling is eliminated.

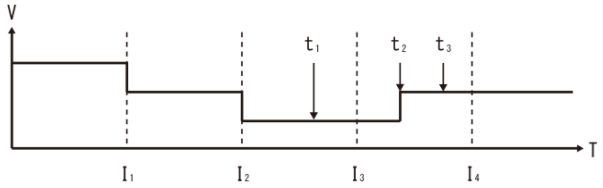

Figure 8. Voltage Scaling with Dual-sensing Canary

VI. Experimental Result

Figure 9 shows execution cycles relative to that of the baseline model. The org_m50 and org_m100 shows *margin50%* and *margin100%* in Figure 4, respectively. dual_m50 and dual_m100 show *margin50%* and *margin100%* in the case where the original canary is replaced with the dual-sensing canary. The execution time is improved to as much as the baseline case in all benchmark programs. The voltage oscillations are reduced. As you can see in Figure 10, EDP is improved to about 80% and 65% of the baseline in *margin50%* and in *margin100%*.

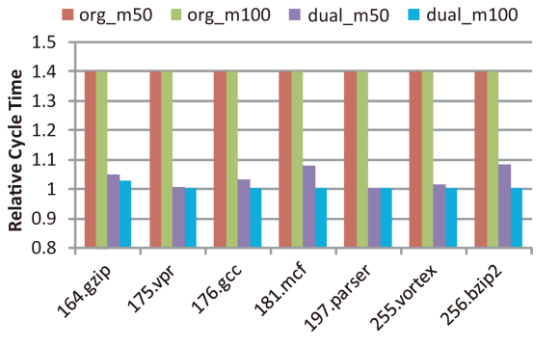

Figure 9. Execution Cycles Improvement via Dual-Sensing Canary

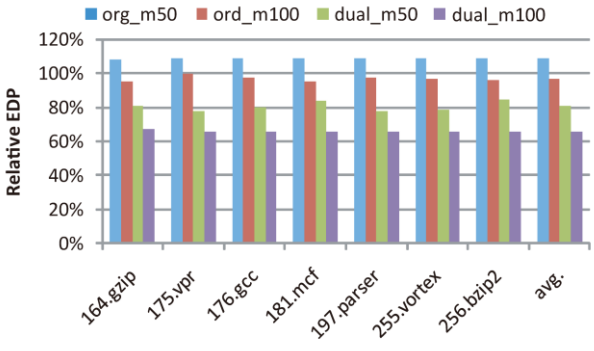

Figure 10. EDP Improvement via Dual-Sensing Canary

VII. Conclusion

In the DVS system, scaling the voltage suffers large performance overhead. Therefore, efficient control of voltage scaling is required. The traditional DVS system with the canary logic has the oscillation in supply voltage, which causes performance loss. In this paper, we introduced dual-sensing canary that is a modification of the canary logic. We changed the voltage scaling policy by utilizing the dual-sensing canary. We found that the performance loss is eliminated with the equivalent energy reduction. Unfortunately, the dual-sensing canary might cause an area overhead because of additional canary FFs. Currently, we are trying to reduce the area overhead by limiting the number of canary FFs.

Acknowledgment

We gratefully acknowledge comments and advices provided by the members of the SoC Laboratory of Kyushu University. Hitachi 0.18µm standard cell libraries are provided by VDEC in the University of Tokyo. This work is partially supported by the CREST program of Japan Science and Technology Agency and by Grant-in-Aid for Scientific Research (KAKENHI) (A) #19200004 and (B) #20300019 from Japan Society for the Promotion of Science.

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V.De, "Parameter Variations and Impact on Circuits and Microarchitecture", ACM/ IEEE Design Automation Conference, 2003.

[2] S. Das, D. Roberts, L. S. Lee, S. Pant, D. Blaauw, T.Austin, K.Flautner, and T.Mudge, "A Self-Tuning DVS Processor Using Delay-Error Detection and Correction", IEEE Journal of Solid-State Circuits, Vol.41, No.4, 2006.

[3] S. Gochman, R. Ronen, I. Anati, A. Berkovits, T. Kurts, A. Naveh, A. Saeed, Z. Sperber, and R. C. Valentine, "The Intel Pentium M Processor: Microarchitecture and Performance", Intel Technology Journal, Vol.7, No.2, 2003.

[4] Intel Corporation, "Intel Pentium M Processor on 90nm Process with 2-MB L2 Cache", Datasheet, 2006.

[5] T. Karnik, S. Borker, and V. De "Sub-90nm Technologes: Challenges and Opportunities for CAD", International Conference on Computer Aided Design, 2002.

[6] E. Larson, S. Chatterjee, and T.Austin, "MASE: A Novel Infrastructure for Detailed Microarchitectural Modeling", International Symposium on Performance Analysis of Systems and Software, 2001.

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

[8] X. Tang,V. K. De, and J. D. Meindl, "Intrinsic MOSFET Parameter Fluctuations Due to Random Dopantplacement", IEEE Transactions on VLSI Systems, Vol.5, No.4, 1997.

[9] O. S. Unsal, J. W. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzales, and O. Ergin, **"**Parameter Variations and Impact on Circuits and Microarchitecture", IEEE Micro, Vol.26, No.6, 2006.