九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Architecture Challenge on SoC Design with Unreliable Transistors

Sato, Toshinori
System LSI Research Center, Kyushu University

https://hdl.handle.net/2324/8312

出版情報：International SoC Design Conference, pp.25-30, 2007-10
バージョン：
権利関係：

Architecture Challenge on SoC Design with Unreliable Transistors

Toshinori Sato
System LSI Research Center
Kyushu University
Fukuoka, Japan
toshinoti.sato@computer.org

Abstract - *The aggressive technology scaling brings us new challenges, such as soft errors, parameter variations, and device wearout. They increase the unreliability of transistors and thus have become a serious problem in SoC designs. Now is the time when circuit and architecture researchers collaborate to attack the problem. The famous WWW Computer Architecture Page says* Computer architecture is the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals*. Every architecture must be built upon some underlying technology available in the realm of engineering. For example, early computers were made of unreliable electron tubes. Architecture still contributes to design SoCs with unreliable components, transistors. There are several techniques proposed from naïve to ambitious, which are based on detection and correction, prediction, and reduction of faults. This paper gives an overview of our recent progress in dependable SoC research.*

Keywords: Soft errors, parameter variations, device wearout, typical-case design, canary flip-flop.

1 Introduction

While microprocessor performance has increased by the advance in semiconductor technology for more than three decades, it has to be further improved to satisfy emerging requirements. An example is security problem. Antivirus program wastes a large portion of processing power, which could be utilized by application programs if our computers were safe from the threat of computer viruses. It is predicted that even smart cell phones will require antivirus as well as PCs in the near future. Another example is mobile multimedia applications. Modern cell phones can play video games, digital still and video cameras, MP3 player, TV phone, and digital TV. Hence, SoCs, which are embedded in mobile devices, require more performance. It looks easy to improve SoC performance, since the advanced semiconductor technologies will provide billions of transistors on a single chip, which enable a super computer on a chip. However this scaling is very difficult due to new challenges such as soft errors, parameter variations, and device wearout. They increase the unreliability of transistors and thus have become a serious problem in SoC designs. To attack the problem, several architectural techniques are proposed from naïve to ambitious. This paper gives an overview of our ongoing research on dependable SoCs.

The rest of the paper is organized as follows. Section 2 explains the challenges in SoC designs and our approaches for them. Sections 3, 4, and 5 introduce multiple clustered core processors, canary flip-flops, and instruction cascading, respectively. Finally, Section 6 closes the paper.

2 Designs with unreliable transistors

As semiconductor technologies have been scaling, new challenges of soft errors, parameter variations, and device wearout are emerging. First, this section explains their impact on SoC designs. After that, it presents our approach on dependable SoC designs.

2.1 Challenges on SoC designs

Increasing soft error rate (SER) is now a major concern for SoC manufactures [8]. With the reduction in transistor size, the area per logic state bit scales down. In order to prevent breakdown caused by high electric field, the supply voltage also scales down. Hence, the node charge reduces and the bit cell is easy to flip by neutrons coming from outer space and alpha particles released by radioactive impurities. This results in logic errors. Since each bit cell becomes small so that probability that some particles hit the cell will also become small, resulting in the net effect of almost constant SER per bit. Since the number of transistors per SoC chip has been tremendously increased following Moore's law, SER per chip is also exponentially increasing.

Variations on an SoC chip are classified into die-to-die (D2D) and within-die (WID) variations. Recently, the latter ones, especially random WID variations, have become serious. Random dopant placement and line edge roughness (Process variations), supply voltage integrity (Voltage variations), and temperature fluctuations (Temperature variations) cause parameter variations [1, 3, 16, 17]. Process variations are essential in semiconductor technologies and they affect each transistor's threshold voltage, resulting in performance variations. Each of PVT variations increases a safety margin, which is required

since delays are no longer constant, and thus SoC designs with considering worst case is becoming very difficult.

Negative Bias Temperature Instability (NBTI) [6] is a dominant wearout mechanism causing an increase of pMOS threshold voltage. This results in the slowdown of transistor switching speed and thus in the circuit performance degradation. In other words, a non-critical path might cause timing violation after aging. Researchers predict that the circuit impact of NBTI will become increasingly significant as semiconductor technologies continue to scale.

These challenges increase the unreliability of transistors and thus have become a serious problem in SoC designs. Next, architecture research focusing on designing SoCs with unreliable transistors is discussed.

2.2 Our approaches for design challenges

Several techniques are proposed to attack the challenges; soft errors, parameter variations, and device wearout. For soft errors, naïve techniques, which are based on detection and correction of faults, are proposed. For parameter variations, typical-case design methodology is proposed. An ambitious technique that aims to reduce faults is also investigated. For device wearout, a prediction-based technique is proposed.

One of the simple implementations for providing soft-error resilience is redundant executions. Time and space redundancies are traditionally utilized. Exploiting time redundancy in a single processor is proposed [9, 10]. Detecting faults due to single event upsets (SEU) is possible by duplicating every instruction in the program. Two redundant copies of the instruction are redundantly executed in the same processor, and two results for the single instruction are compared with each other. If they do not match, an SEU is detected. Instruction reissue mechanisms are utilized to duplicate every instruction. This is based on detection and correction of faults.

Multiple clustered core processor (MCCP), which exploits space redundancy to detect SEUs [12, 13], is also proposed. MCCP is described in Section 3.

Parameter variations will make the traditional worst-case design impossible, since they can not provide design margins that it requires. Considering the situation, design methodology has to be changed. Typical-case design methodology [5] is a promising one. It exploits an observation that worst cases are rare. Designers should focus on typical cases rather than worst cases. Since worst cases do not have to be considered, design constraints are relaxed, resulting in easy designs. Constructive Timing Violation (CTV) technique [5, 11] and canary flip-flop (FF) [14, 15] are example realizations of the typical-case design methodology.

The concept of CTV is as follows. Every timing critical function in an SoC chip is designed by two methods. The design consists of two components. One is called main part, and the other is called checker part. While two parts share the single function, their roles and implementations are mutually different. The main part is designed with performance considerations, but might cause timing errors. The checker part is a safety net for the main part. It detects timing errors that occur in the main part, and thus it has to satisfy all timing constrains in the chip. However, designers do not have to optimize performance nor power but only have to guarantee the function. If a timing error is detected by the checker part, the circuit state has to be recovered to a safe point. That is, CTV is based on detection and correction of faults.

Canary FF predicts timing errors. It consists of two FFs, a delay element, and a comparator. The FFs, which are called main FF and shadow FF respectively, hold an input value redundantly. The delay element is placed between the previous logic stage and the shadow FF, and the shadow FF might cause timing errors even when the main FF does not. When two values held in the FFs do not match, a timing error is predicted. Canary FF is described in Section 4.

Instruction cascading is an ambitious technique that aims to reduce faults. It exploits the statistical characteristics of circuit delay [2]; the mean delay increases and the standard deviation decreases as the number of independent critical paths increases, and both the mean delay and the standard deviation decrease as the logic depth increases. Since it is expected that the decrease in the standard deviation would reduce faults due to variations, instruction cascading increases the logic depth of the execution stage. Instruction cascading is described in Section 5.

Canary FF can also be utilized for wearout prediction [15]. This is because there are no difference between timing error due to variations and the one due to wearout.

As mentioned above, Sections 3, 4, and 5 present the detail explanations of MCCP, canary FF, and instruction cascading, respectively.

3 Multiple clustered core processor

Figure 1 shows an MCCP. While it is a homogeneous multicore processor, it consists of multiple clustered cores rather than monolithic ones. Each core is based on the clustered microarchitecture [4]. MCCP shown in Figure 1 consists of two homogeneous clustered cores, each of which has two identical clusters. Each

cluster has its dedicated instruction scheduling queue (IQ), register files (RF), and functional units (FU). Instruction and data caches (I$ and D$), branch predictor (BrPred) and decoder (Decode) are shared by the clusters.

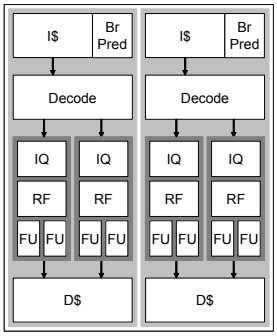

Figure 1. Multiple clustered core processor

MCCPs have a good characteristic in its dependability. One of the simple implementations for providing dependability is to redundantly execute a single program. The conventional multicore processors can exploit space redundancy for dependability [7]. In order to check errorless, a single thread is redundantly executed on multiple cores. When two outcomes for the single thread (or those for every instruction in the thread) do not match, an SEU is detected. Furthermore, an MCCP can change its dependability mode according to the importance of the current thread, as shown in Figure 2 [12, 13]. If the thread is critical, it is duplicated and redundantly executed across multiple cores, as shown in Figure 2(a). If it is less critical, every instruction in the thread is duplicated and redundantly executed across multiple clusters, as shown in Figure 2(b).

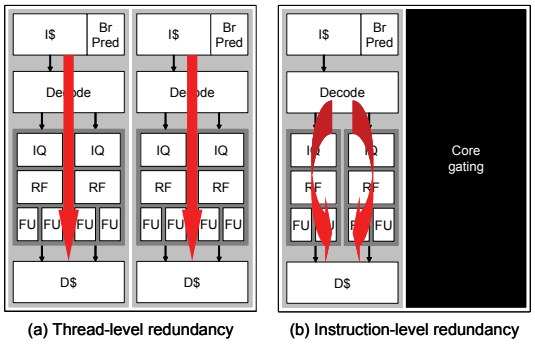

Figure 2. Dependability mode

The amount of instruction level parallelism (ILP) varies between application programs. Thread-level redundancy is exploited only when ILP is high, and similarly instruction-level redundancy is exploited only when ILP is low, as sown in Figure 3 [13]. When ILP is low, there are idle execution resources and thus dependability is provided without serious performance loss. In addition, since a core is occasionally turned off, the wasted power consumption is eliminated.

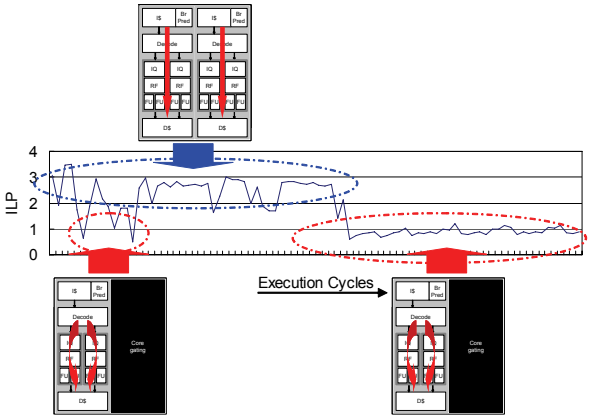

Figure 3. ILP-directed mode selection

Figure 4 shows performance and energy-delay product (EDP) of an MCCP that relies on the ILP-directed mode selection [13]. Instructions per cycle (IPC) is used as a metric for evaluating performance. Line graphs present IPC and bar graphs present EDP. Every value is normalized by the corresponding value of an MCCP that always relies upon thread-level redundancy. It is observed that IPC is reduced by 13% on average, while EDP is improved by 13% on average.

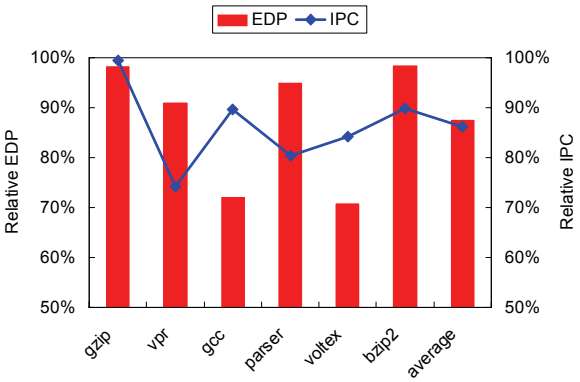

Figure 4. Simulation results (MCCP)

4 Canary flip-flop

Canary FF is an FF, whose design is augmented with a delay element and a shadow FF, as shown in Figure 5. It is used as a canary in a coal mine to help predict whether a timing error is about to occur. Timing error prediction is based on the comparison between the main FF value and that of the shadow FF, which runs into the timing error a little bit before the main FF. An alert signal triggers voltage control.

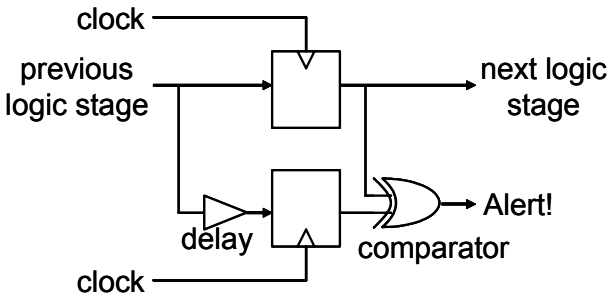

Figure 5. Canary flip-flop

Currently, canary FF is investigated for eliminating power consumed due to the safety margin. Generally, processor's maximum clock frequency is determined by considering the worst-case critical-path delay and the safety margin. The margin is required since delays are not constant due to parameter variations. In the worst-case design, the margins are summed up as shown in Figure 6 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.

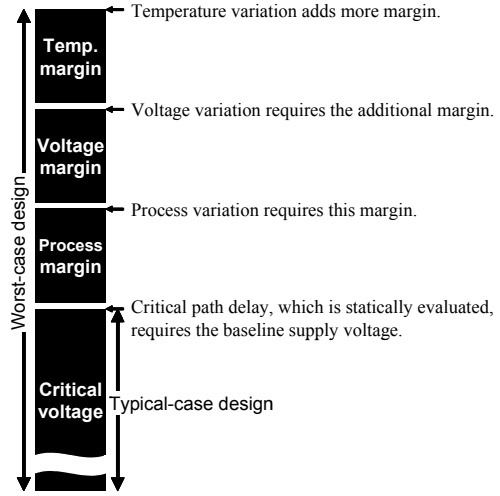

Figure 6. Safety margins

An adaptable dynamic voltage scaling (DVS) technique is utilized to scale the supply voltage to the point where every margin described above is eliminated. This allows supply voltage reduction, resulting in significant energy savings. Figure 7 explains how DVS technique utilizes 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. This is possible since input values activating the critical path are limited to a few variations. Timing errors rarely occur even if the timing constraints on the critical path are not satisfied. The input value variations can be exploited to decrease the supply voltage. Because the supply voltage is lower than that determined by the critical path delay, significant power reduction is expected. When a timing error is predicted to occur, the supply voltage is increased to the next higher voltage as shown in Figure 7.

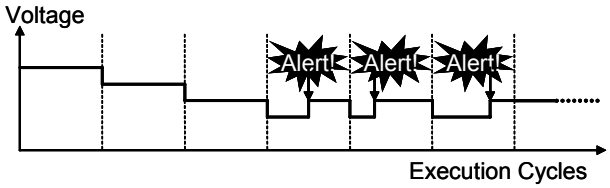

Figure 7. DVS system

Figure 8 shows processor performance and EDP of a processor where the DVS technique is utilized [15]. Line graphs present IPC and bar graphs present EDP. Every value is normalized by the corresponding value of the processor that does not utilize the DVS system. It is observed that EDP is improved by approximately 10% on average without severe performance loss. Because it is not well-known how large safety margin every variation requires, input value variations rather than PVT variations are considered. Further energy reduction must be expected when design margins required by PVT variations are eliminated.

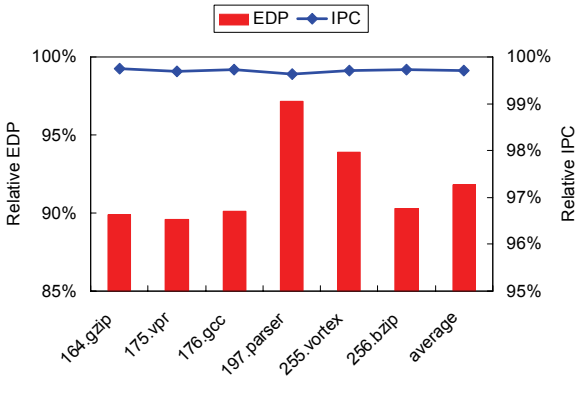

Figure 8. Simulation results (Canary FF)

5 Instruction cascading

Based on the study by Hashimoto et al. [2], we are investigating variation resilient microarchitectures. They demonstrate that due to the statistical characteristics of delays (1) the mean delay increases and the standard deviation decreases as the number of independent logic critical paths increases, and (2) both the mean delay and the standard deviation decrease as the logic depth increases. That means variation resilient microarchitectures prefer shallow pipeline depth. Next, we investigate how to increase the logic depth in a pipeline stage.

In order to increase the logic depth in the execution stage, we are considering the use of instruction cascading (or instruction collapsing [18]). This paper focuses on the backend of microprocessors. It remains for the future study how to increase the logic depth in the frontend of processors. Instruction cascading is a microarchitectural technique that collapses several instructions and executes them in a single cycle, as shown in Figure 9. In this example, two dependent instructions are cascaded and then executed on a cascaded ALU (or interlock collapsing ALU [18]) in one cycle.

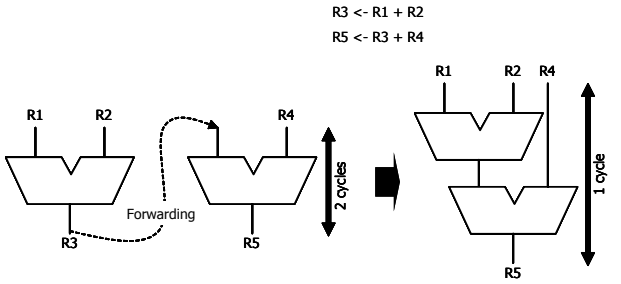

Figure 9. Instruction cascading

Following the example used in [2], it is explained how instruction cascading improves variation resilience. Suppose there are 100 logic critical paths in the conventional ALU. Also suppose the delay of each path is distributed according to a normal distribution $N(6, 1^2)$; the mean delay is 6 and the standard deviation is 1. The bold line (denoted as N(6,1)x100) in Figure 10 illustrates the critical path delay distribution of the ALU. If two instructions are collapsed into a cascade, the distribution changes into the thin line (denoted as Cascaded2). Here, it is assumed that the mean delay increases into 12, that the standard deviation changes into $2^{1/2}$, and that the number of logic critical path decreases into 50. In addition, considering that instruction throughput is two times increased, the effective delay per operation is calculated by dividing the original delay by two. Similarly, the dashed line (denoted as Cascaded4) shows the logic critical path delay distribution of the cascaded ALU, which collapses four instructions. As can be easily seen, both the effective mean delay and the standard deviation decrease as the number of cascaded instructions increases. Hence, we find that instruction cascading is beneficial for improving variation resilience.

Traditionally, instruction cascading is used in order to improve processor performance, and thus clock frequency is maintained even when the complex cascaded ALU is included in a microprocessor. Since the focus of this paper is in variation resilience rather than in performance improvement, slower clock frequency should be selected when the cascaded ALU is used. Even so, it is expected that processor performance per second would be maintained if instruction throughput were increased by the cascading. It is expected that only cascadable instructions are on the program critical path. In contrast, uncascadable instructions are not on the program critical path and thus they do not have any impact on the execution time of the program. That is, it is not a serious problem that there are a noticeable number of uncascadable instructions. The consideration behind this expectation is as follows. Since uncascadable instructions do not make a chain of instructions, they can not construct any program critical path. Therefore, uncascadable instructions do not have serious impact on processor performance and thus processor performance per second would be maintained.

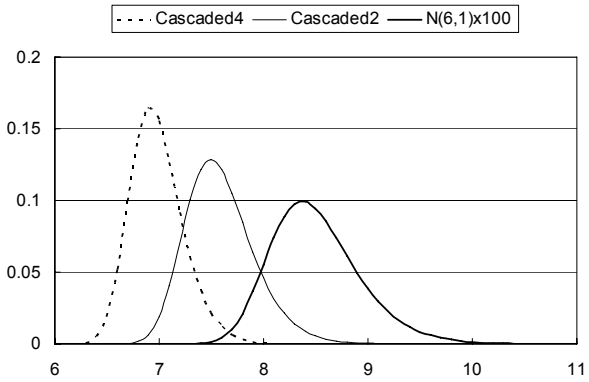

Figure 10. Critical path delay distributions

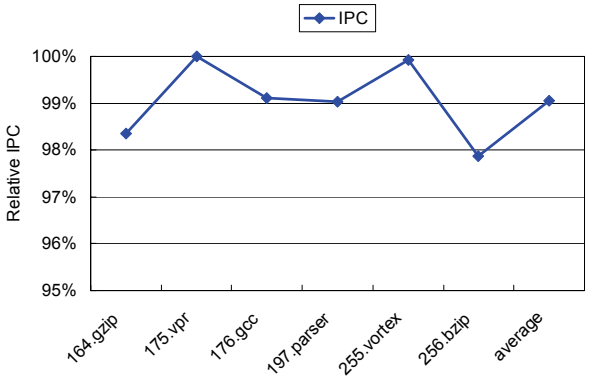

Figure 11. Simulation results (Cascading)

Figure 11 presents a very preliminary simulation results. In order to confirm that uncascadable instructions do not have any serious impact on performance, the execution latency of such instructions is dynamically increased in simulations. The latency of cascadable instructions is unchanged. Note that if processor performance is maintained even if the latency of uncascadable instruction is increased, the expectation is confirmed. Figure 11 shows performance of processor that utilizes instruction cascading for variation resilience. Every value is normalized by that of the conventional processor. It can be easily observed that the decrease in

IPC is very small; an average of 1.0% and up to 2.1%. Hence, we confirm the expectation above is appropriate; uncascadable instructions do not have serious impact on processor performance and thus instruction cascading is a promising variation-resilient microarchitecture.

6 Conclusions

As scaling in semiconductor technologies continues, SoC designs will face with new challenges; soft errors, parameter variations, and device wearout. In order to design SoCs with unreliable transistors, several techniques are proposed. This paper surveyed our latest achievements in dependable SoC research.

Acknowledgements

The ongoing work is being done in collaboration with Toshimasa Funaki, Yuji Kunitake, and Shingo Watanabe of Kyushu Institute of Technology. The author gratefully acknowledges advice and help provided by the members of the SoC laboratory of Kyushu University. This work is partially supported by the CREST (Core Research for Evolutional Science and Technology) program of Japan Science and Technology Agency (JST) and by Grant-in-Aid for Scientific Research (KAKENHI) (A) #19200004 from Japan Society for the Promotion of Science (JSPS).

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter Variations and Impact on Circuits and Microarchitecture," 40th IEEE/ACM Design Automation Conference, 2003.

[2] M. Hashimoto and H. Onodera, "Increase in Delay Uncertainty by Performance Optimization," IEEE International Symposium on Circuits and Systems, 2001.

[3] T. Karnik, S. Borkar, and V. De, "Sub-90nm Technologies: Challenges and Opportunities for CAD," 20th IEEE/ACM International Conference on Computer Aided Design, 2002.

[4] R. E. Kessler, "The Alpha 21264 Microprocessor," *IEEE Micro*, Vol. 19, No. 2, 1999.

[5] Y. Kunitake, A. Chiyonobu, K. Tanaka, and T. Sato, "Challenges in Evaluations for a Typical-Case Design Methodology," 8th IEEE International Symposium on Quality Electronic Design, 2007.

[6] V. Reddy, A. T. Krishnan, A. Marshall, J. Rodriguez, S. Natarajan, T. Rost, and S. Krishnan, "Impact of Negative Bias Temperature Instability on Digital Circuit Reliability," 40th IEEE International Reliability Physics Symposium, 2002.

[7] E. Rotenberg, "AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors," 29th IEEE International Symposium on Fault-Tolerant Computing, 1999.

[8] P. I. Rubinfeld, "Managing Problems at High Speed," *IEEE Computer*, Vol. 31, No. 1, 1998.

[9] T. Sato, "Exploiting Instruction Redundancy for Transient Fault Tolerance," 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2003.

[10] T. Sato and I. Arita, "In Search of Efficient Reliable Processor Design," 30th International Conference on Parallel Processing, 2001.

[11] T. Sato and I. Arita, "Constructive Timing Violation for Improving Energy Efficiency," L. Benini, M. Kandemir, J. Ramanujam (Eds.), *Compilers and Operating Systems for Low Power*, Kluwer Academic Publishers, 2003.

[12] T. Sato and A. Chiyonobu, "Multiple Clustered Core Processors," 13th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2006.

[13] T. Sato and T. Funaki, "Power-Performance Trade-off of a Dependable Multicore Processor," 13th IEEE Pacific Rim International Symposium on Dependable Computing, 2007.

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

[15] T. Sato and Y. Kunitake, "Critical Issues Regarding A Variation Resilient Flip-Flop," 14th Workshop on Synthesis and System Integration of Mixed Information Technologies, 2007.

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

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

[18] S. Vassiliadis, J. Phillips, and B. Blaner, "Interlock Collapsing ALU's," *IEEE Transactions on Computers*, Vol. 42, No. 7, 1993.