Constructive Timing Violation for Improving Energy Efficiency

Toshinori Sato[1,2] Itsujiro Arita[1]
[1] Department of Artificial Intelligence
[2] Center for Microelectronic Systems
Kyushu Institute of Technology, Japan
{tsato,arita}@ai.kyutech.ac.jp

Abstract

A novel technique to improve energy efficiency is disclosed. It relies on a fault-tolerance mechanism for timing constraints based on speculative execution technique. Since power reduces quadratically with supply voltage, supply voltage reduction can result in substantial power savings. However, it also causes larger gate delay, and thus clock must be slow down in order not to violate timing constraints of critical paths. If any fault-tolerance mechanism is provided for timing faults, it is not necessary to keep the constraints. From these observations, we propose a fault-tolerance technique for timing failures, which efficiently utilizes the speculative execution mechanism and reduces power consumption. We call the technique constructive timing violation. *This paper evaluates our proposal using a cycle-by-cycle simulator and finds its efficiency on energy consumption.*
Keywords: low power design, fault tolerance, timing constraints, branch prediction, speculative execution

1 Introduction

Current trend of increasing popularity of portable and mobile computer platforms such as laptop and cell phone is a driving force to investigate high-performance and power-efficient microprocessors. For example, Java-2 MicroEdition (J2ME) works on cell phones[7]. We can download a game and play it on our cell phone. Travellers guide and flight ticket reservation are available. Furthermore, mobile banking and trading are also provided. As computing power of mobile device increases, its power consumption is also increasing. The active power P_{active} and gate delay t_{pd} of a CMOS circuit are given by

$$P_{active} = fC_{load}V_{dd}^2 \quad (1)$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \quad (2)$$

where f is clock frequency, C_{load} is load capacitance, V_{dd} is supply voltage, and V_{th} is threshold voltage of the device. α is a factor depending upon the carrier velocity saturation and is about 1.3–1.5 in advanced MOSFETs[4]. Based on Eq.(1), it is easily found that power supply reduction is the most effective way to lower power consumption. However, Eq.(2) tells us that supply voltage reduction increases gate delay, resulting in slower clock frequency. And thus, computing performance of microprocessor is diminished.

It is possible not to degrade computing power by maintaining clock frequency before and after supply voltage reduction. This causes timing faults, resulting in logic errors. However, if any fault-tolerance mechanism is provided for the faults, the logic errors can be avoided. In other words, we propose to give up meeting timing constraints but to tolerate violations[10]. We call this technique *constructive timing violation* (CTV) and have already applied it for boosting computing power. In this paper, we evaluate it on the research area of low power design.

2 Low Power via Fault-Tolerance

Our proposal is based on a kind of parallelism, that is space redundancy. An element of circuits consists of a main part and checker parts. The main part is responsible for computing power, but it could suffer timing faults. The checker parts, which are timing error free, support the main part and revert processor state to a safe point where a timing fault is detected. The main and checker parts are equivalent but their clock frequencies are different with each other. To the main part, we provide clock frequency which is higher than that decided by the critical path, since it is expected that typical delay of the circuit is less than critical delay and that timing faults rarely occur[6]. On the other hand, since the checker parts are used for detecting the timing faults, they work at the frequency which meets the critical delay. This design technique exploits the fact that the longest path for an individual operation of a logic circuit is generally much shorter than a critical path of the circuit. Furthermore, it utilizes the fact that input signals which decide the critical path are limited to a few variations. Considering the characteristics of logic circuits and their critical paths, the circuits could be designed as the longest path decided by most of the operations for a function is shorter than an expected cycle time.

The power reduction is achieved as follows. It is assumed that the main part is error free when the voltage of V_{dd} is supplied. Based on Eq.(2), its maximum clock frequency f_{dd} is as follows.

$$f_{dd} \propto \frac{(V_{dd} - V_{th})^{1.3-1.5}}{V_{dd}} \tag{3}$$

For easy understanding, Eq.(3) is simplified as

$$f_{dd} \propto V_{dd} \tag{4}$$

without loss of generality. In order to reduce power consumption, we would like to supply the voltage of V_L lower than V_{dd}. Usually, the clock frequency should be reduced to f_L determined by V_L, but we keep it as f_{dd}. The checker parts are used for detecting the timing faults and thus they work at the frequency f_L with the supply voltage V_L. That is, they are timing error free. In order to maintain throughput, the checker parts are duplicated if necessary. If power reduction due to the lower supply voltage is larger than increase of power consumption caused by amount of parallelism, the proposed technique can decrease power consumption efficiently.

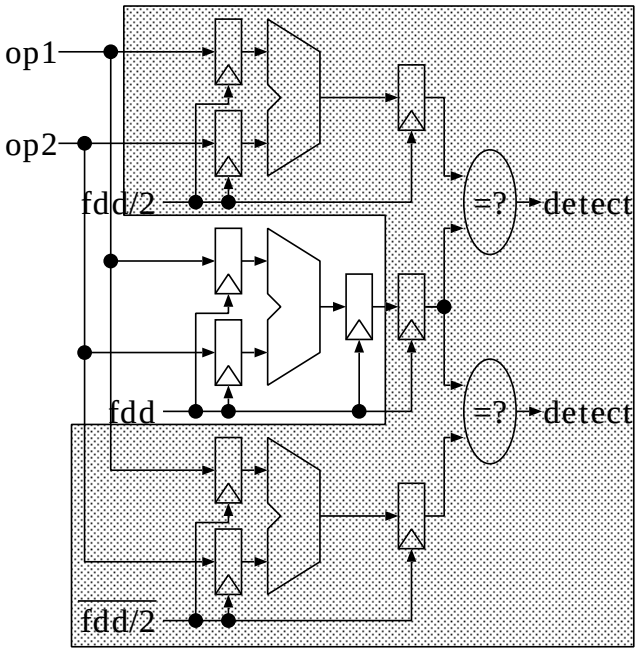


Figure 1. ALU utilizing proposed technique

We explain our proposal using an example. However, this is applicable to any combinational logics. Figure 1 depicts an ALU which utilizes the proposed technique. The shaded box shows additional circuits for fault-tolerance. It is assumed that currently the ALU executes at the clock frequency f_{dd} with the supply voltage V_{dd}, and it is expected that the supply voltage is reduced to V_L, say $V_L = \frac{V_{dd}}{2}$. First, the ALU is duplicated by three times. One ALU, which we call main ALU, works at f_{dd} with V_L and the remaining two ALUs, which we call checker ALUs, work at $f_L = \frac{f_{dd}}{2}$ with V_L. Thus, the checker ALUs are free from timing errors and are used for verifying the operation of the main ALU. Please note that the main ALU is essential for maintaining low latency and that the checker ALUs only maintain throughput. If there are serious dependences between instructions, high throughput can not be maintained

without the main ALU. Using this technique, the ALU's power consumption is reduced as follows. The power in the main ALU is reduced from $f_{dd}C_{load}V_{dd}^2$ to $f_{dd}C_{load}(\frac{V_{dd}}{2})^2 = \frac{f_{dd}C_{load}V_{dd}^2}{4}$. The power in two checker ALUs is $2 * \frac{f_{dd}}{2}C_{load}(\frac{V_{dd}}{2})^2 = \frac{f_{dd}C_{load}V_{dd}^2}{4}$. Thus, total power consumption is reduced from $f_{dd}C_{load}V_{dd}^2$ to $\frac{f_{dd}C_{load}V_{dd}^2}{2}$. While this represents rather ideal case, the power reduction is realistic and attractive when we use typical value of 1.3–1.5 for α. The aim of this estimate is demonstrating the potential of the CTV.

Please note again that this approach is applicable not only to datapath such as ALU but also to any combinational logics. One of the applications of this approach for control path is the logic detecting data dependences between instructions. This logic is on one of the critical paths in in-order issue microprocessors. Assuming this logic is asserted and is critical only when there are dependences, the CTV mechanism can relief timing constraints so as not to cause timing violations only if there are not any dependences between instructions.

Clock signals distributed to the ALUs are shown in Figure 2. Since the clock signals of the checker ALUs are complementary with each other, they work alternatively to verify the main ALU. Figure 2 explains how two consecutive operations start and are verified. The verification is based on comparing two outputs from the main ALU and corresponding one of the checker ALUs. If they do not match, a timing fault is detected. In such cases, any recovery action should be initiated. Since the comparators should be fault free, they will work at slower clock frequency f_L. In order to revert processor state to a safe point where the error is detected, we propose to utilize the recovery mechanism used in modern microprocessors for speculative execution. In other words, a timing fault of an instruction is regarded as a misspeculated instruction. Thus, there are no hardware overhead in the recovery mechanism.

We consider two mechanisms for the recovery action. One uses the existing speculation recovery mechanism for mispredicted branches, and the other is based on the instruction reissue mechanism for incorrect data speculation[8, 9], which will be included in future microprocessors[5]. Note that the correct value is provided by the checker ALUs, and thus instruction retry is successful for recovery from timing faults. That is, when a fault is detected, it is enough to re-execute instructions following the fault instruction. In the case that we utilize the recovery mechanism for mispredicted branches, when a timing fault is detected, the microprocessor flushes its pipeline and then restarts at the corresponding instruction. All instructions following the fault instruction are squashed and thus penalty might be very large. We call this recovery mechanism instruction squashing. On the other hand, using the instruction reissue, only instructions dependent upon the fault instruction are selectively invalidated and re-executed. Hence, low performance loss is expected. Explaining the process of the instruction reissue is beyond the scope of this paper and can be found in [9].

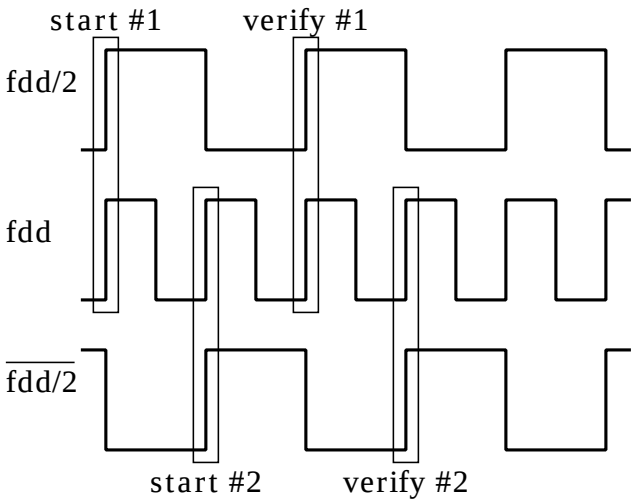


Figure 2. Clock signals

From these considerations, it is possible to detect timing faults and to tolerate the violations.

3 Evaluation Methodology

We implemented a timing simulator using SimpleScalar/Alpha tool set (ver.3.0a)[2]. The baseline model is an out-of-order execution superscalar processor based on the register update unit[11],

Table 1. Processor configuration

Fetch Width	4 instructions
Branch Predictor	512-set, 4-way set-associative BTB, 2048-entry bimodal predictor, updated in commit stage, 8-entry return address stack, 3-cycle miss penalty
Insn. Windows	16-entry instruction queue, 8-entry load/store queue
Issue Width	4 instructions
Commit Width	4 instructions
Functional Units	4 iALU's, 1 iMUL/DIV, 2 Ld/St's, 4 fALU's, 1 fMUL/DIV
Latency(total/issue)	iALU 1/1, iMUL 3/1, iDIV 20/19, Ld/St 2/1, fADD 2/1, fMUL 4/1, fDIV 12/12
Register Files	32 32-bit fixed point registers, 32 32-bit floating point registers
Insn. Cache	16K direct-mapped, 32-byte blocks, 6-cycle miss penalty
Data Cache	16K 4-way set-associative, 32-byte blocks, 2-port, write-back, non-blocking load, hit under miss, 6-cycle miss penalty
L2 Cache	unified, 256K 4-way set-associative, 64-byte blocks, 48-cycle miss penalty

Table 2. Benchmark programs

program	input set
164.gzip	input.compressed
175.vpr	net.in arch.in
176.gcc	cccp.i
186.crafty	crafty.in
197.parser	test.in
252.eon	chair
255.vortex	lendian.raw
256.bzip2	input.random

and its configuration is summarized in Table 1.

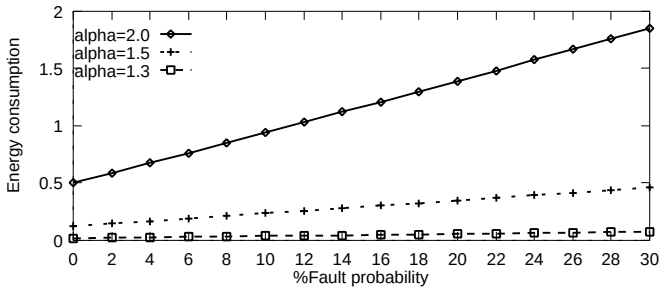
The SPEC2000 CINT benchmark suite is used for this study. Table 2 lists the benchmarks and the input sets. We use the object files provided by University of Michigan. For each program except for `252.eon`, 1 billion instructions are skipped before actual simulation begin. Each program is executed to completion or for 100 million instructions. We do not count nop instructions.

4 Simulation Results

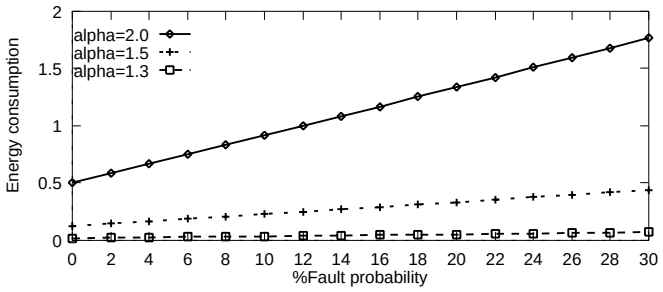
In this section, we present preliminary results using the approach described in Section 2. Note that the technique should be applied to every element which probably violates timing constraints. When we use one main part and two checker parts, the clock frequencies f_L should satisfy the following condition.

$$\frac{1}{2} f_{dd} \leq f_L < f_{dd}$$

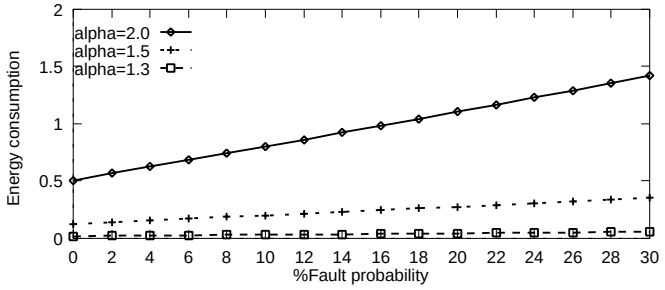
Thus, we choose the clock frequency of $\frac{1}{2} f_{dd}$ for the checker parts since efficiency of power reduction is largest. We vary probability, that timing faults occur, randomly between 0 and 30% of operations, and measure energy consumption. Note that in practice there must be correlation between fault probability and f_L. Deciding the optimal tradeoff point is remained for the future study. Figure 3 presents energy consumption when the instruction squashing is used. It is calculated by multiplying active power by execution cycles, and is normalized by that of the baseline model. Even in the case that α equals 2.0, the energy consumption is substantially reduced if the fault probability is less than approximately 15%. The rising energy consumption is caused by increasing cycle time due to recovery from timing errors. Since the instruction squashing involves long miss penalty for the recovery, high fault probability diminishes power efficiency considerably. This loss will be mitigated



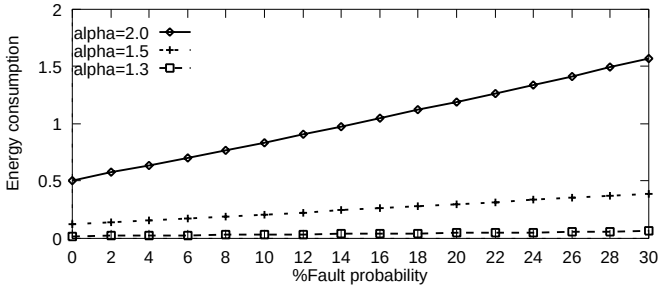
(i) 164.gzip



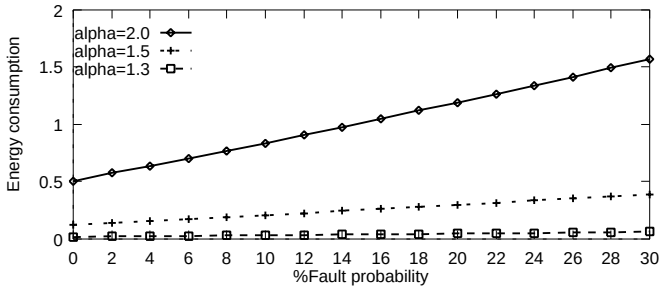
(ii) 175.vpr



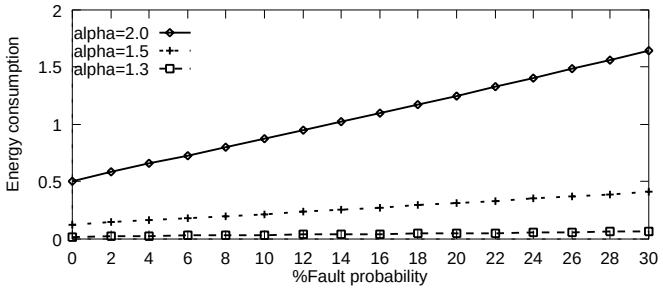
(iii) 176.gcc



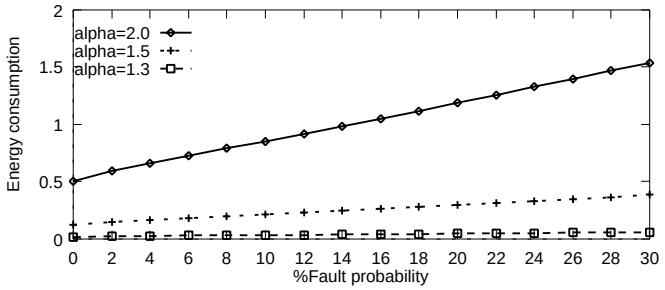
(iv) 186.crafty



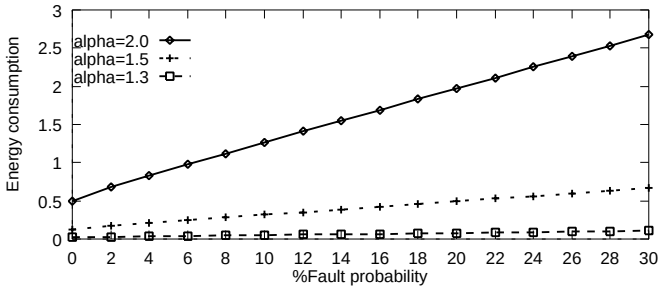
(v) 197.parser



(vi) 252.eon

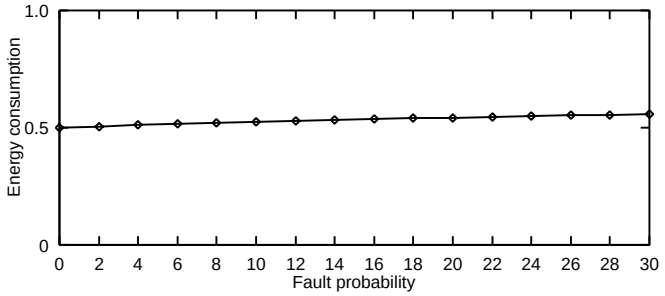


(vii) 255.vortex

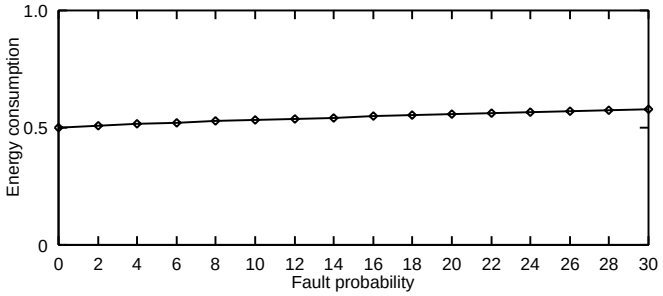


(viii) 256.bzip2

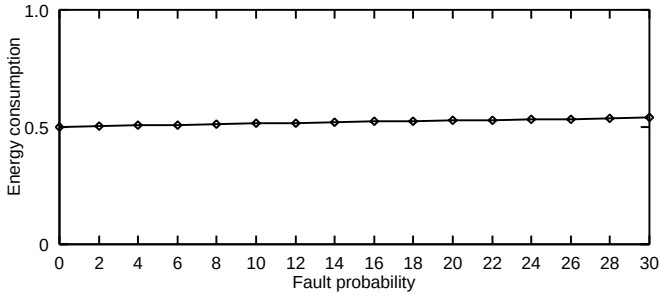
Figure 3. Energy consumption (Squash)



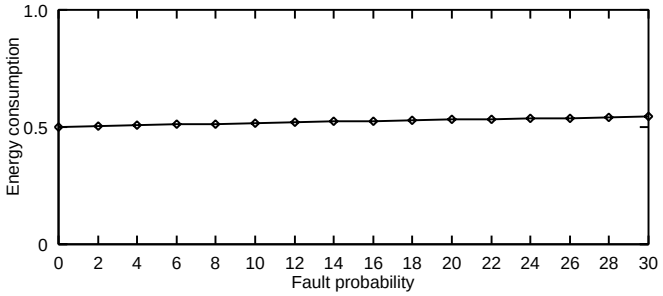
(i) 164.gzip



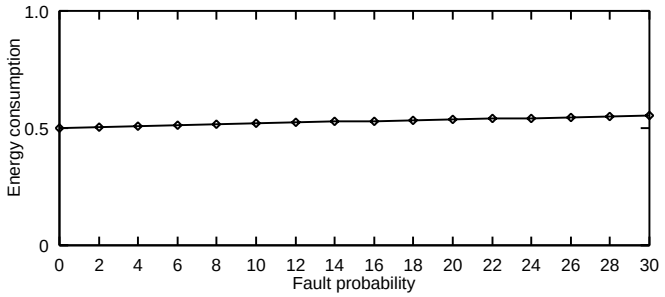
(ii) 175.vpr



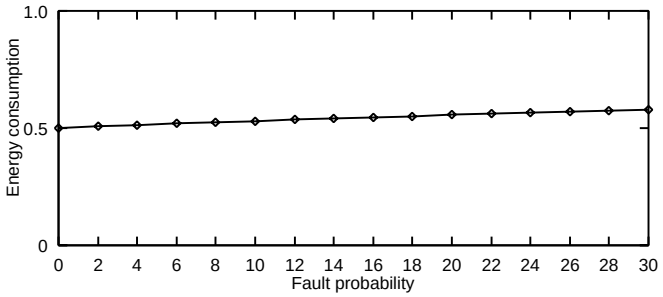
(iii) 176.gcc



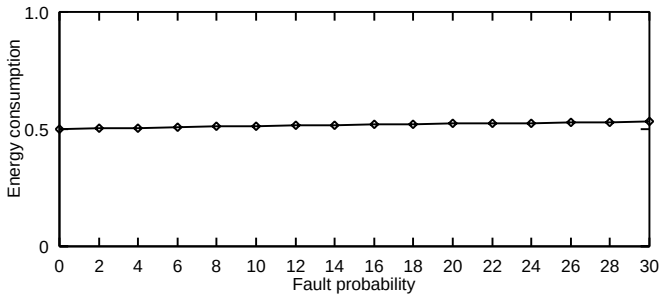
(iv) 186.crafty



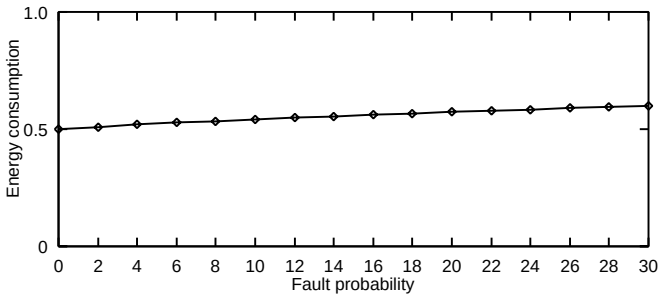
(v) 197.parser



(vi) 252.eon



(vii) 255.vortex



(viii) 256.bzip2

Figure 4. Energy consumption (Reissue)

by small miss penalty of the instruction reissue. If we use typical value of 1.3–1.5 for α, we can observe that the energy consumption is significantly reduced even when the fault probability reaches 30%.

Figure 4 presents energy consumption when the instruction reissue is used. Only the case where α equals 2.0 is shown. As can be easily seen, the energy reduction of 50% is almost maintained across the fault frequencies evaluated. While this technique requires slightly high-cost instruction reissue mechanism, it will be implemented in future microprocessors[5]. Thus, it is applicable for mobile PCs to utilize this fault-tolerance technique for reducing energy consumption.

5 Related Work

Kondo et al.[6] propose Variable Latency Pipeline (VLP) structure for integer ALUs. Using properly two kinds of circuits according to the longest path of the circuits for each operation, the effective execution latency can be almost one cycle. Our proposal is strongly influenced by the VLP, while they are not interested in fault-tolerance nor power-efficiency at all. In addition, our proposal is applicable not only to ALUs but also to any combinational logics. Using SPEC92 CINT benchmark suite, the usefulness of the VLP is evaluated on a platform of an in-order execution scalar processor, but it is not clear for dynamically-scheduled superscalar processors. In this paper, we made our evaluation on the 4-way dynamically-scheduled superscalar processor.

Chandrakasan et al.[3] utilize a parallel architecture to reduce supply voltage with maintaining throughput. Two identical circuits are used in order to make each unit to work at half the original frequency while the original throughput is maintained. Since the speed requirement for the circuit becomes half, the supply voltage can be decreased. In this case, the amount of parallelism can be increased to further reduce the total power consumption. Our proposal also utilize parallelism. However, we maintain not only throughput but also latency of the applied element. Thus, our proposal is efficient even for irregular applications which do not have much data parallelism.

DIVA[1] is an example of a fault-tolerant microprocessor based on space redundancy. A simple checker processor is used for dynamically verifying committed instructions. Any hardware faults are corrected using recovery mechanism for incorrect branch predictions. Hence, DIVA is a hardware-based mechanism and transparent. However, DIVA requires additional ports for register files and caches in order for the checker processor to share processor contexts. This increases design complexity and circuit delay of its main processor. In addition, power-efficiency is not considered.

6 Conclusion and Future Work

In this paper, we proposed a fault-tolerance technique to improve energy efficiency of microprocessors and named it *constructive timing violation*. The preliminary evaluation shows that the proposed mechanism is energy-efficient under the condition that timing errors occur infrequently.

One of the future studies is modeling relationship between clock frequency and fault probability. The model is useful to understand the effectiveness of the proposed technique in real world. The other future direction is reducing hardware overhead for introducing fault-tolerance. One of the possible solutions is sharing circuits between the main and checker parts by pipelining the element using transparent latches. This eliminates the duplication of each element and thus reduces hardware budget considerably.

Acknowledgement

This work is supported in part by a financial gift from Toshiba Corporation.

References

[1] T.M.Austin, "DIVA: a reliable substrate for deep submicron microarchitecture design," 32nd International Symposium on Microarchitecture, 1999.

[2] D.Burger and T.M.Austin, "The SimpleScalar tool set, version 2.0," ACM

SIGARCH Computer Architecture News, vol.25, no.3, 1997.

[3] A.P. Chandrakasan and R.W. Brodersen, "Minimizing power consumption in digital CMOS circuits," Proceedings of IEEE, vol.83, no.4, 1995.

[4] T.Hiramoto and M.Takamiya, "Low power and low voltage MOSFETs with variable threshold voltage controlled by back-bias," IEICE Transactions on Electronics, vol.E83-C, no.2, 2000.

[5] Intel Corporation, "Inside the NetBurst micro-architecture of the Intel Pentium 4 processor," White paper, 2000.

[6] Y.Kondo, N.Ikumi, K.Ueno, J.Mori, and M.Hirano, "An early-completion-detecting ALU for a 1GHz 64b datapath," International Solid State Circuit Conference, 1997.

[7] M.Levy, "Java to go: part 1," Microprocessor Report, vol.15, archive 2, 2001.

[8] M.H.Lipasti, C.B.Wilkerson, and J.P.Shen, "Value locality and load value prediction," International Conference on Architectural Support for Programming Languages and Operating Systems VII, 1996.

[9] T.Sato, "Data dependence speculation using data address prediction and its enhancement with instruction reissue," 24th Euromicro Conference, Workshop on Digital System Design: Architectures, Methods and Tools, 1998.

[10] T.Sato and I.Arita, "Give up meeting timing constraints, but tolerate violations," Cool Chips IV, 2001.

[11] G. S. Sohi, "Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers," IEEE Transactions on Computers, vol.39, no.3, 1990.