Evaluating the Potential of an Energy Reduction Technique Based on Timing Constraint Speculation

Asami Tanino

Department of Artificial Intelligence
Kyushu Institute of Technology
asami@mickey.ai.kyutech.ac.jp

Toshinori Sato

Department of Artificial Intelligence
Kyushu Institute of Technology
toshinori.sato@computer.org

Abstract

Recently, we proposed a technique improving energy efficiency named Constructive Timing Violation *(CTV)[17]. In this paper, in order to evaluate the practicability of the CTV, we design Carry Select Adder (CSLA) using Verilog-HDL, and investigates the distribution of timing failures depending on timing constraints. It is found that the CVT improves energy efficiency by over 40% while the fault probability is 50% when clock frequency is 2 times faster than that satisfies timing constraints due to critical paths.*
Keywords: timing constraints, speculative execution, fault tolerance

1 Introduction

Currently, smart mobile devices and embedded systems require high computing capability, thus employing high performance microprocessors. In addition, however, they require low power consumption as well as high performance. Because there is a tradeoff between power consumption and performance in microprocessors, power is the primary design constraint in embedded microprocessors for mobile devices. The active power P_{active} and gate delay t_{pd} of a CMOS circuit are given by

$$P_{active} \propto f C_{load} V_{dd}^2 \quad (1)$$

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

where f is the clock frequency, C_{load} is the load capacitance, V_{dd} is the supply voltage, and is V_{th} the threshold voltage of the device. α is a factor dependent upon the carrier velocity saturation and is approximately 1.3–1.5 in advanced MOSFETs[5]. Based on Eq.(1), it can easily be found that a power-supply reduction is the most effective way to lower power consumption. However, Eq.(2) tells us that reductions in the supply voltage increase gate delay, resulting in a slower clock frequency, and thus diminishing the computing performance of the microprocessor.

It is possible to not degrade computing power by maintaining clock frequency before and after reductions in supply voltage. This approach causes timing violations, however, resulting in logic errors. If a fault-tolerance mechanism is provided for the violations, however, the logic errors can be avoided. In other words, to improve energy efficiency we propose that timing constraints not be met and that violations be tolerated[17]. We call this technique *constructive timing violation* (CTV) and have already applied it to boosting computing power[16]. In this paper, we evaluate this new technique with regard to the research area of low-power design.

2 Constructive Timing Violation

Our proposal is based on a kind of parallelism, that is a space redundancy. Circuits consist of a main part and checker parts. The main part is responsible for computing power —high through-

put and low latency—, but it can suffer timing violations. In contrast, the checker parts, which are free of timing violations, support the main part and revert the processor state to a safe point where a timing violation is detected. The main and checker parts are equivalent in design, but their distribution of clock frequencies is different. For the main part, we provide a clock frequency that is higher than that decided by the critical path, since it is expected that a typical circuit delay is shorter than the critical delay and that timing faults rarely occur[7]. For example, it has been reported that nearly 80% of paths have delays of half the critical time[21]. On the other hand, since the checker parts are used for detecting the timing violations, they work at a frequency that 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 the critical path of the circuit. Furthermore, it utilizes the input signals, which determine the critical path, being limited to a few variations. Considering the characteristics of logic circuits and their critical paths, the circuits could be designed as the longest path determined by most of the operations, for a function is shorter than the expected cycle time. In order to maintain throughput, the checker parts are duplicated if necessary. Please note that the main part is essential for maintaining low latency and that the checker parts only maintain throughput. If there are serious dependencies between instructions, high throughput also cannot be maintained without the main part.

The power reduction is achieved as follows. It is assumed that the main part is violation-free when the V_{dd} voltage 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: i.e. it is assumed that α is 2. On the suffice this simplification is fairly impractical, especially when V_{dd} is small. However, it is already known that t_{pd} is not sensitive to V_{th} scaling if V_{th} is scaled together with V_{dd}[3]. We expect that this joint scaling will be possible with future technologies such as VT-CMOS[8] and ABB-MTCMOS[12]. In addition, V_{dd} reduction in deep-submicron CMOS is not as detrimental to speed as in long-channel CMOS[3]. To reduce power consumption, we would like to supply a voltage V_L lower than V_{dd}. This is possible by using on-chip DC/DC converters[15]. The clock frequency should usually be reduced to f_L, as determined by V_L, but we keep it as f_{dd}. The checker parts are used to detect timing violations and thus they work at a frequency f_L with the supply voltage V_L. That is, they are timing-violation free. If the power reduction due to the lower supply voltage is larger than the increase in power consumption caused by the amount of parallelism, the proposed technique can efficiently decrease power consumption.

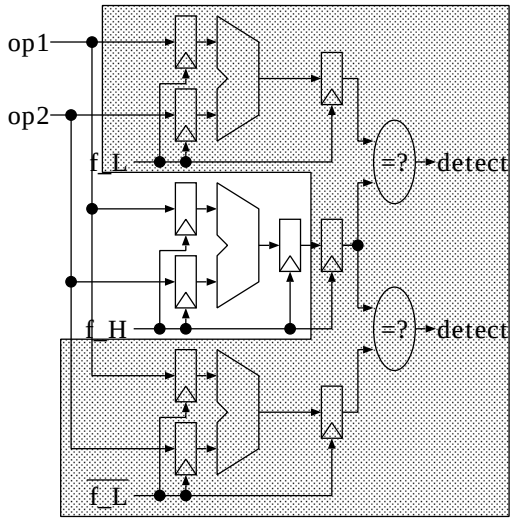

Figure 1. ALU utilizing proposed technique

We will explain our proposal using an example. However, this proposal is applicable to any combinational logics, including datapaths and control logics. The only stipulation is that latches should meet timing constraints. In other words, timing violations should not occur in latches when working at a higher clock that is distributed to

the main part. Figure 1 depicts an ALU utilizing the proposed technique. While it is true that cases caches are critical in many and determine clock frequency of a microprocessor, it has been reported that adders and instruction-scheduling mechanisms become bottlenecks limiting the increase in clock frequency[10, 14]. For example, Compaq's Alpha 21264 processor splits execution pipes into two clusters to meet timing constraints, while it has a data cache operating at twice the frequency of the processor clock[6]. Thus, we believe that ALU is suitable for explaining our proposal because of its simplicity. The shaded box shows additional circuits for the checker parts. It is assumed that the ALU currently executes at the clock frequency f_{dd} with 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, ALU is duplicated three times. One ALU, which we call the 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 violations and are used to verify operation of the main ALU. Please remember that the main ALU is essential for maintaining low latency and that the checker ALUs only maintain throughput. If there are serious dependencies between instructions, high throughput cannot 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 the 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 $\frac{f_{dd}C_{load}V_{dd}^2}{2}$. That is, it is reduced to ideally by half. While this represents a rather ideal case, the power reduction is realistic and attractive when we use typical values of 1.3–1.5 for α. The aim of this estimate is to demonstrate CTV potential.

It should be explained again that this approach is applicable not only to a datapath such as ALU but also to any combinational logics. One of the applications of this approach for the control path is the logic that detects data dependencies 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 dependencies, the CTV mechanism can relieve timing constraints so as not to cause timing violations only if there are no dependencies between instructions.

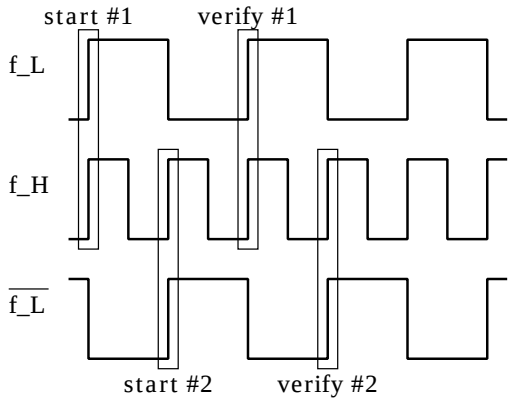

Figure 2. Clock signals

Clock signals distributed to the ALUs are shown in Figure 2. Because the clock signals of the checker ALUs are complementary, 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 one of the corresponding checker ALUs. If the outputs do not match, a timing violation is detected. In such cases, a recovery action should be initiated. Since the comparators should be violation-free, they will work at a slower clock frequency f_L. In order to revert the processor state to a safe point where the error is detected, we propose utilization of a recovery mechanism used in modern microprocessors for speculative execution. In other words, an instruction's timing violation is regarded as a misspeculated instruction. Thus, there is little hardware overhead in the recovery mechanism.

We consider a mechanism for the recovery action. It is based on the instruction-reissue mechanism for incorrect data speculation[9, 18]. Its limited mechanism has already been included in modern microprocessors[4, 6]. Note that the correct value is provided by the checker ALUs, and

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

thus instruction retry is successful for recovery from timing faults. That is, when a fault is detected, it is sufficient to re-execute instructions following the fault instruction. When using instruction-reissue, only instructions dependent upon the fault instruction are selectively invalidated and re-executed. Hence, little performance loss is expected. Explaining the process of instruction-reissue is beyond the scope of this paper and can be found in [18].

Based on these considerations, it is possible to detect timing faults and to tolerate violations.

3 Potential of CTV

The goal of this section is not convincing that the CTV is practical, but presenting its potential on improving energy efficiency. First, we describe our evaluation environment. After that, we will show simulation results.

3.1 Evaluation environment

We implemented a cycle-by-cycle 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[19], 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. They were compiled by DEC C V5.9-008 on Digital UNIX V4.0 (Rev.1229). For each program, 1 billion instructions are skipped before actual simulation begins. Each program is executed to completion or for 100 million instructions. We do not count nop instructions.

Table 2. Benchmark programs

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

When we use one main part and two checker parts, the clock frequencies f_L and f_H should sat-

isfy the following condition.

$$f_L \leq f_H < 2 * f_L$$

In this evaluation, we choose the clock frequencies of $f_H = 2 * f_L$ and f_L for the main and checker parts respectively. In order to evaluate how timing violations affect energy efficiency, we make them occur randomly. We vary probability that timing violations occur between 0 and 50% of operations, and measure processor performance. We call the probability *fault probability*.

3.2 Results

In this section, we present preliminary results using the approach described in Section 2. Please note that the technique should be applied to every element which probably violates timing constraints. For measuring performance, we use the committed instructions per second (IPS). Only useful instructions are considered for counting the IPS. In other words, nop instructions or those flushed by branch mispredictions or timing violations are not included when we calculate the IPS.

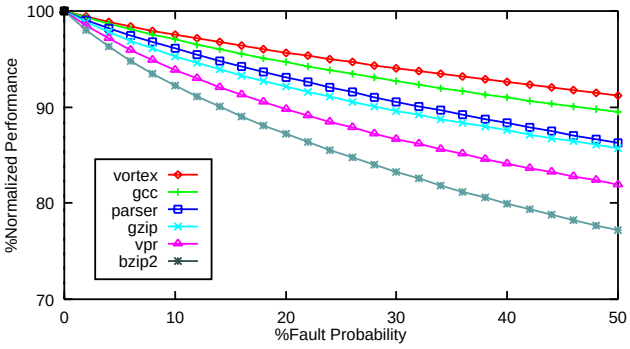

Figure 3. Processor performance

Figure 3 presents processor performance when the CTV is utilized. Performance of the evaluated model is normalized by that of the baseline model, which is explained in Table 1. Figure 3 shows that processor performance is monotonically degraded as the fault probability is increased. However, in some programs, the performance degradation is less than 10% even when the fault probability is 50%.

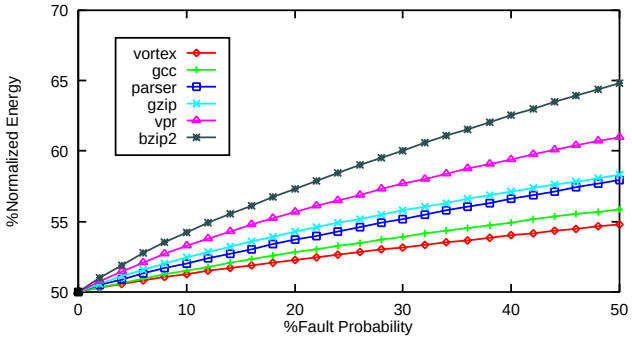

Figure 4. Energy consumption

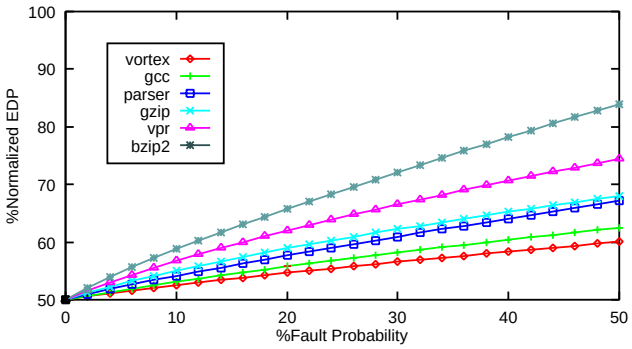

Figure 5. Energy-delay product

Figures 4 and 5 present energy consumption and energy-delay product (EDP) when the CTV is utilized. Energy consumption and EDP of the evaluated model are normalized by those of the baseline model. Thus, if in the figures they mark less than 100%, we can find that energy efficiency is improved. Figure 5 shows that energy efficiency is diminished as the fault probability is increased. However, in most cases, EDP is reduced by more than 30%.

4 A Case Study: CSLA

This section presents a case study on evaluating practicability of the CTV. In order to model the relationship between boosted clock frequency and timing error rate, we implement CSLA[13]. Matsuo et al. evaluated the similar technique as the CTV on Carry Lookahead Adder[11].

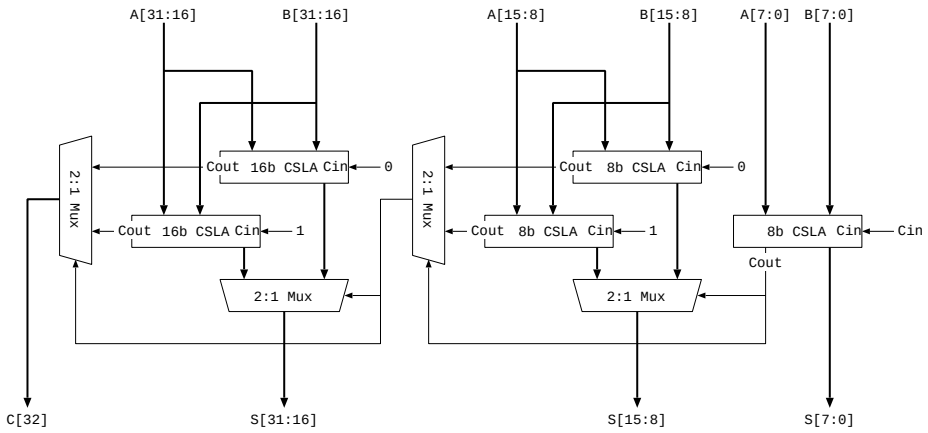

Figure 6. Carry select adder

4.1 Evaluation environment

We implement our CSLA using Verilog-HDL due to the lack of transistor-level process technology information. It is shown in Figure 6. After verifying the correctness of its function by logic simulation, we logic-synthesize it using Synopsis DesignCompiler. The synthesized CSLA includes estimated gate and wire delay based on a standard ASIC library.

In order to detect timing violation, we simulate two CSLAs simultaneously on Cadence Verilog-XL, as shown in Figure 7. One is the upper CSLA with considering delay, which is the gate-level circuit synthesized by DesignCompiler. The other is the lower CSLA without considering delay, which is the RTL description that we implemented using Verilog-HDL. As increasing clock frequency distributed to the registers, the comparator signals there is a difference between results from the two CSLAs.

In order to perform logic simulation, we make test vectors from functional simulation using SPEC2000 CINT benchmark listed in Table 2. Due to large simulation time, we restrict test vectors only 40 thousand patterns for each program.

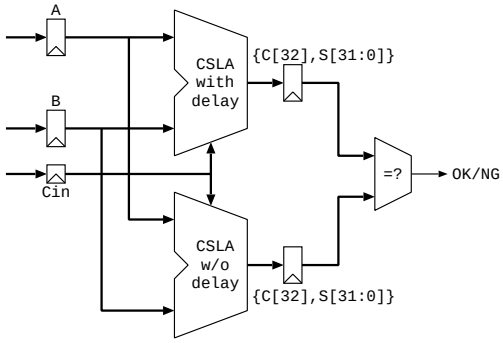

Figure 7. Evaluation circuit

4.2 Results

Figure 8 shows logic simulation results, when we boosted processor clock frequency by the order of 1.1 – 3.0. The horizontal line indicates the boosting ratio and the vertical line indicates the fault probability. We can find that fault probability is less than 50% when clock frequency is 2 times faster than that satisfies timing constraints due to critical paths. For example, in the case of `176.gcc`, the fault probability is approximately 10% when $f_H = 2 * f_L$. Figure 5 shows that energy efficiency is improved by over 40% when the fault probability is 10%.

5 Related Work

Kondo et al.[7] 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

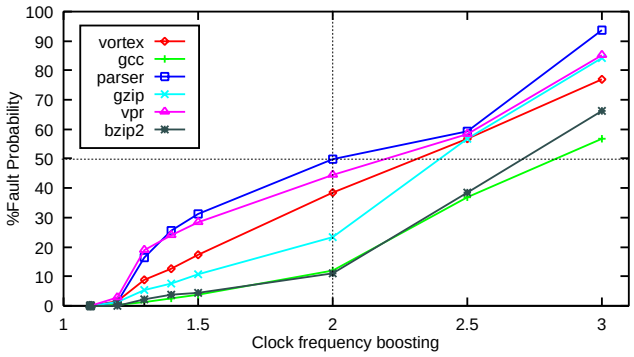

Figure 8. Fault probability

effective execution latency can be almost one cycle while its critical path is longer than one cycle. Our proposal is strongly influenced by the VLP. However, 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 single-issue processor, but it is not clear for dynamically-scheduled multiple-issue processors. In contrast, we made our evaluation on the out-of-order execution 4-way superscalar processor.

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. One of the problems on DIVA is that it requires additional ports for register files and caches in order for the checker processor to share processor contexts with the main processor. This increases design complexity and circuit delay of its main processor.

6 Open Issues and Future Work

Currently, we evaluated practicability only on ALUs. We should evaluate whole processors. One of the major problems is cache memories. It is easily expected that they always occur timing violations if its clock frequency is boosted. We need a breakthrough to apply the CTV to caches. One possible solution is giving up boosting clock frequency distributed to caches. This increases cache access latency and might have performance loss. Thus, it is required to evaluate detailed simulations when the CTV is applied to the remaining blocks other than cache.

Duplicating a circuit to tolerate timing violation is a very simple method. In addition, the area required for the circuit such as ALU is negligible in the current semiconductor technology. However, the duplicating might change floorplaning. This causes timing violations in other blocks in the microprocessor. One of the solutions is coarse grain CTV. That is, the whole microprocessor is duplicated. This maintains the floorplaning, however, some recovery mechanism other than that used in this paper is required for violations.

The benefit of the CVT would be determined when it is used on a whole design such as a microprocessor or other application specific designs using Verilog-HDL and synthesized gates. Further, it is better to evaluate CSLA with layout and spice simulations.

Currently, we are considering to condense multiple ALU operations[4, 22]. When we utilize the CTV for the condensing, it becomes possible to execute dependent operations in one cycle by dynamically identifying small delay operations. We expect that ALU condensing has some contribution to processor performance. This gain in performance can be translated into energy reduction.

We are also interested in combining the CTV with the slipstream processor[20]. This is the coarse grain CTV. The advanced stream is executed in the main processor, and the redundant stream is executed in the checker processor. We expected that only one checker processor is required in the coarse grain CTV, because the slipstream processor improves performance of the redundant stream by supplying execution results of the advanced stream to the redundant one as predictions.

7 Conclusion

In this paper, we proposed the constructive timing-violation (CTV) for improving energy efficiency. Reducing supply voltage causes timing

violations, resulting in logic errors. However, if any tolerant mechanism for the timing violations is provided, these errors can be avoided. We explained the example mechanism which is based on space redundancy and modern speculative execution. Every timing violation is detected by the checker circuits and can be recovered by the misprediction recovery mechanism.

Using cycle-by-cycle simulations, we have found that the CTV efficiently improves energy efficiency. When clock frequency is boosted by the factor of 2, the CTV improves processor performance for all cases even when the fault probability equals 50%. The boosting clock frequency by 100% improves energy efficiency by over 40%.

In order to evaluate practicability of the CTV, we implemented CSLA and investigate its fault probability. We found that fault probability is less than 50% when clock frequency is 2 times faster than that satisfies timing constraints due to critical paths. Because 50% of the fault probability did not have severe impact on energy efficiency, processors may tolerate the measured 50% of the fault probability, and it is confirmed that the CTV technique is effective on improving energy efficiency.

Acknowledgement

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

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] K. Chen and C. Hu, "Performance and V_{dd} scaling in deep submicrometer CMOS," IEEE Journal of Solid-State Circuits, vol.33, no.10, 1998.

[4] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel, "The microarchitecture of the Pentium 4 processor," Intel Technical Journal, issue Q1, 2001.

[5] 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.

[6] R. E. Kessler, E. J. McLellan, and D. A. Webb, "The Alpha 21264 microprocessor architecture," International Conference on Computer Design, 1998.

[7] 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.

[8] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, F. Sano, M. Norishima, M. Murota, M. Kato, M. Kinugasa, M. Kakumu, and T. Sakurai, "A 0.9V, 150MHz, 10mW, 4mm^2, 2-D discrete cosine transform core processor with variable-threshold-voltage scheme," International Solid State Circuit Conference, 1996.

[9] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value locality and load value prediction," 7th International Conference on Architectural Support for Programming Languages and Operating Systems, 1996.

[10] T. Liu and S-L. Lu, "Performance improvement with circuit-level speculation," 33rd International Symposium on Microarchitecture, 2000.

[11] T. Matsuo, T. Fujikawa, K. Metsugu, and K. Murakami, "Dependable pipelining: microarchitecture for the multi-GHz generation," Technical Report of IEICE, DC2002-20, 2002 (in Japanese).

[12] K. Nii, H. Makino, Y. Tujihashi, C. Morishima, and Y. Hayakawa, "A low power

SRAM using auto-backgate-controlled MT-CMOS," International Symposium on Low Power Electronics and Design, 1998.

[13] V.G.Oklobdzija, "High-speed VLSI arithmetic units: adders and multipliers," in A. Chandrakasan, W. J. Bowhill, and F. Fox, "Design of high-performance microprocessor circuits," IEEE Press, 2001.

[14] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," 24th International Symposium on Computer Architecture, 1997.

[15] S. Sakiyama, J. Kajiwara, M. Kinoshita, K. Satomi, K. Ohtani, and A. Matsuzawa, "An on-chip high-efficiency and low-noise DC/DC converter using divided switches with current control technique," International Solid-State Circuits Conference, 1999.

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

[17] T. Sato and I. Arita, "Constructive timing violation for improving energy efficiency," 2nd Workshop on Compilers and Operating Systems for Low Power, 2001.

[18] T.Sato, "Evaluating the impact of reissued instructions on data speculative processor performance," Microprocessors and Microsystems, vol.25, issue 9-10, 2002.

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

[20] K. Sundaramoorthy, Z. Purser, and E. Rotenberg, "Slipstream processors: improving both performance and fault tolerance," 9th International Conference on Architectural Support for Programming Languages and Operating Systems, 2000.

[21] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanazawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," IEEE Journal of Solid-State Circuits, vol.33, no.3, 1998.

[22] S. Vassiliadis and J. Phillips: "Interlock collapsing ALUs," IEEE Transactions on Computers, vol.42,nNo.7, 1993.