Simultaneous Dynamic and Static Power Reduction Utilizing Power Heterogeneous Functional Units

Yoshimi Shibata Takanori Hayashida Toshinori Sato* Shinya Takahashi
Department of Electronics Engineering and Computer Sciences
Fukuoka University, Fukuoka 814-0180 Japan
Tel: +81-92-871-6631, Fax: +81-92-865-6031
E-mail: *toshinori.sato@computer.org

***Abstract*— Power consumption is the major constraint for modern microprocessor designs. Especially, static power consumption becomes a serious problem as transistor size is shrunk via semiconductor technology improvement. This paper proposes a technique that reduces static power consumed by functional units. It exploits activity rate of functional units and utilizes power heterogeneous functional units. From detailed simulations, we investigate the conditions where the proposed technique works well for simultaneous dynamic and static power reduction.**

I. INTRODUCTION

As semiconductor technologies have aggressively improved, power consumption due to leakage current becomes one of the serious problems in microprocessor designs. The static power is always consumed regardless of circuit activity and thus tremendously increases as the number of transistors on a single chip increases. Several techniques are proposed to eliminate or to reduce the static power consumption [1].

Sleep transistor [1, 2] is a virtual power switch, which is inserted into the target circuit in series. When the sleep transistor is off, the power supply is virtually cut off and thus the static power is almost eliminated. The issue to consider in adoption of sleep transistor is break even time (BET) [2]. Because the power switch consumes dynamic power to discharge and charge the capacitance of the target circuit. It also requires the additional circuit that controls the power switch, which also consumes the additional power. BET is a time period where the power saving by cutting of the power supply is equal to the total overhead mentioned above. The period when the power supply is cut off should be longer than BET to reduce power and it is considerably long. Ikebuchi et al. propose to predict when functional units in an in-order processor are inactive and to turn off the power supply only when the period is predicted to be longer than BET [2].

While it is easy for in-order processors to predict when each functional unit is active, it is almost impossible for out-of-order processors. This is because the functional unit is selected just before the instruction is ready to execute. There is not enough time to predict. Recently, even embedded systems adopt out-of-order processors. For example, both Apple's A5 [3] and NVIDIA's Tegra-3 [4] SoCs integrate ARM Cortes-A9 core [5], which is an out-of-order processor. These SoCs are widely used in smart phones and tablet PCs, which desire high performance and low power consumption. Hence, a new technique to reduce static power consumed in out-of-order processors is strongly required.

II. CMOS POWER CONSUMPTION

CMOS power consumption is governed by the equation:

$$P = P_{dynamic} + P_{static}$$

where $P_{dynamic}$ is the dynamic power and P_{static} is the static power. $P_{dynamic}$ and gate delay t_{pd} are given by

$$P_{dynamic} \propto f \cdot C_{load} \cdot V_{dd}^2$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_t)^{\alpha}}$$

where f is the clock frequency, C_{load} is the load capacitance, V_{dd} is the supply voltage, and V_t is the threshold voltage of the device. α is a factor dependent upon the carrier velocity saturation and is approximately 1.3 - 1.5. It can easily be found that a power supply reduction is the most effective way to lower power consumption. However, reduction in the supply voltage increases gate delay, resulting in less performance in the microprocessor. In order to maintain high transistor switching speeds, it is required that the threshold voltage is proportionally scaled down with the supply voltage.

On the other hand, the static power can be given by

$$P_{static} = I_{static} \cdot V_{dd}$$

where I_{static} is leakage current. The major part of the leakage current is subthreshold leakage current and is dominated by threshold voltage V_t in the following equation:

$$I_{static} \propto 10^{-V_t / S}$$

where S is the subthreshold swing parameter.

Thus, lower threshold voltage leads to increased subthreshold leakage current and increased static power. Maintaining high transistor switching speeds via low threshold voltage gives rise to a significant amount of static power consumption. In other words, it is very difficult to reduce both dynamic and static power with maintaining high performance.

III. POWER HETEROGENEOUS FUNCTIONAL UNITS

Figure 1 shows a superscalar processor core. Every instruction is fetched from the instruction cache (I$). It is decoded in the decoder (Decoder) and then is dispatched into

the instruction window (Instruction Window), where it is waiting until its operands are ready. When the instruction is ready to execute, it is issued into one of the functional units (SDU's and LSU) in an out-of-order fashion. SDU's work for arithmetic and logical operations, while LSU works for load and store operations, which refer the data cache (D$). From the discussions in Section II, we know high-performance processor cores should use leaky transistors. In this paper, we call the functional unit made of leaky transistor *static power dominated unit* (SDU). Even when the instruction finishes, it stays in Instruction Window until it moves into the head of the window. And last, its execution result is written back into the register file (Register File) in an in-order fashion.

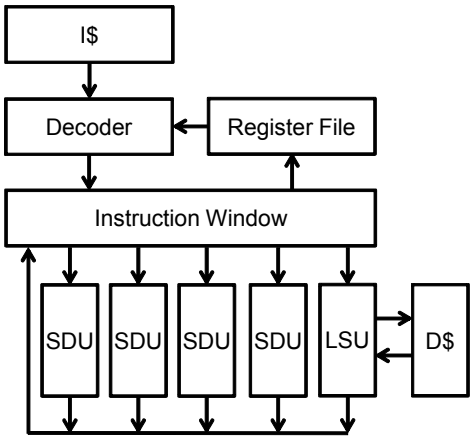

Fig. 1. Superscalar Processor Core

Modern superscalar processor cores have multiple SDU's. Every SDU consumes considerably large static power regardless of its activity. Here, we can exploit activity rate of SDU's. It is widely known that instruction level parallelism is limited and thus every SDU is not always active. In most period of program execution, instruction level parallelism is around 2. This means that about half of the SDU's are inactive, if the processor core has four SDU's as shown in Figure 1. When the number of instructions that execute in parallel is less than that of the SDU's implemented in the core, the selection logic in the instruction window determines into which SDU each one of the instructions is issued. Since this selection happens in a static priority order [6], the highest-priority SDU will always be active. On the other hand, the lowest-priority SDU will be active only in the much rarer case that the full issue width is used. This policy results in the highest-priority SDU being accessed frequently while the lowest-priority ALU is rarely accessed. Hence, we classify SDU's into two classes; SDU's with high activity rate and those with low activity rate. Power consumed by the former ones is dominated by the dynamic power and that consumed by the latter ones is dominated by the static power.

Considering the above observations, we propose to replace the SDU's with low activity rate with the functional units made of leakless transistor. This reduces static power consumed by the less active functional units. However, unfortunately, the threshold voltage of leakless transistor is high and thus the functional unit becomes slow. In order to compensate the speed, we follow [7] and increase its power supply voltage. Though this increase the dynamic power of the unit, it might not matter because its activity rate is low. Here, we call the functional unit that is made of leakless transistor and works at high supply voltage *dynamic power dominated units* (DDU). Note that the supply voltage for SDU does not change. Figure 2 shows the superscalar processor core, two of whose SDU's are replaced by DDU's. We can see heterogeneity in power among the functional units. This technique simultaneously optimizes dynamic and static power consumption and thus reduces total power consumption.

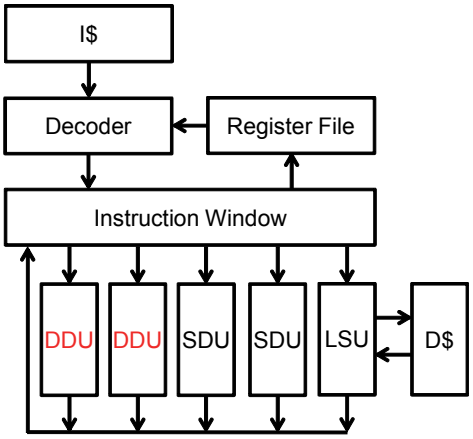

Fig. 2. Processor Core with Power Heterogeneous Functional Units

IV. EVALUATIONS

A. Methodology

SimpleScalar tool set [8] is used for cycle-by-cycle simulation. We selected six benchmark programs from SPEC2000 suite; 164.gzip, 175.vpr, 176.gcc, 197.parser, 255.vortex, and 256.bzip2. For each program, 100M instructions are executed in details. Table 1 summarizes the processor core configuration.

Table 1. Processor Core Configuration

Fetch width	8 instructions
L1 instruction cache	16KB, 2-way, 1-cycle access
Branch predictor	gshare + bimodal
gshare predictor	4K entries, 12 histories
bimodal predictor	4K entries
Branch target buffer	1K set, 4-way
Dispatch width	4 instructions
Instruction queue size	32 instructions + 16-entry LSU
Issue width	4 instructions
Integer ALU's (SDU/DDU)	4 units
Integer multipliers	2 units
Floating-point ALU	1 unit
Floating-point multiplier	1 unit
L1 data cache ports	2 ports
L1 data cache	16KB, 4-way, 2-cycle access
Unified L2 cache	8MB, 8-way, 10-cycle access
Memory	Infinite, 100-cycle access
Commit width	4 instructions

The configurations of the heterogeneous functional units are as follows. First, the processor has four functional units. The sum of the numbers of SDU's and of DDU's is four. Second, we assume that static power consumed by DDU is negligible. This is because dynamic power dominates. Third, we evaluate three scenarios for SDU. We assume the cases where the ratio of the static power over the total power is 30%, 40%, and 50% when all functional units are SDU. This assumption means it is a high-performance processor core. In these cases, the dynamic power of SDU is calculated as 7, 4.5, and 3 times larger than the static power of SDU, respectively.

Under these assumptions, we investigate how the dynamic power of DDU should be. We use the arbitrary unit for measuring power consumption.

B. Results

First, we present the break even point of the proposed technique in Figure 3. The horizontal axis indicates the configurations of the heterogeneous functional units. For example, "DDU:1 SDU:3" means the processor core has one DDU and three SDU's. The vertical axis indicates the active power of DDU. Each one of three lines corresponds to one of three scenarios and presents the boarder where the total power consumption is reduced. If the active power of DDU is under the line, the total power is successfully reduced. Hence, they indicate the break even power (BEP) of DDU. For example, in the case of 50% scenario, the dynamic power of DDU should be less than 11.15, which is 3.72 times larger than that of SDU.

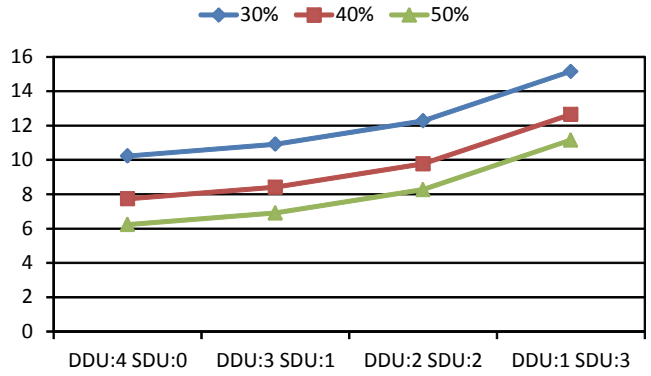

Fig. 3. Break Even Point for DDU Dynamic Power

Figure 4 presents power consumption in the case where the ratio of the static power over the total power is 50% and the dynamic power of DDU is twice larger than that of SDU. The horizontal axe indicates benchmark programs and the vertical axe indicates relative power that is compared with the baseline power consumption. Hence, if the bar is lower than 100%, power consumption is reduced. For each group, four bars from left to right indicate the result when 4, 3, 2, and 1 of SDU's are replaced by DDU, respectively. We can see bathtub curve, and benchmark programs are categorized into 3 types. The first one consists of bzip2 and gzip. In this type, power consumption is monotonously increases as the number of SDU's increases. This means static power dominates. The second one consists of parser and vpr. In this type, on the contrary, power consumption decreases as the number of SDU's increases. This means dynamic power dominates. Figure 5 presents the breakdown of the baseline power consumption. The left part indicates the dynamic power and the right one indicates the static power. They confirm the above discussions. The static power dominates in the first type and the dynamic power dominates in the second type. The third type consists of gcc and vortex and locates between the two types. Unfortunately, in the cases of parser and vpr, the proposed technique does not achieve power reduction. However, on average, the total power consumption is reduced by 3.76%, 9.13%, 11.2%, and 8.20%, respectively, for the cases where 4, 3, 2, and 1 of SDU's are replaced by DDU's.

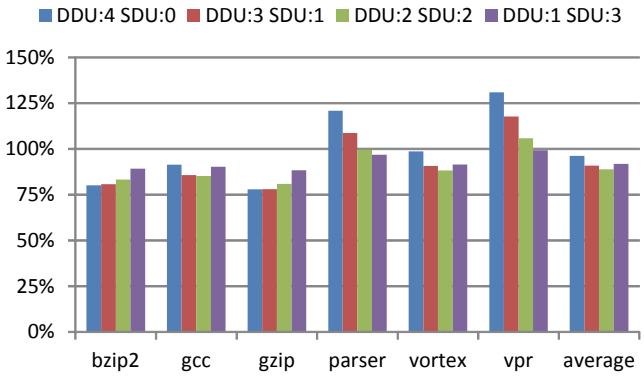

Fig. 4. Relative Power Consumption (50%-scenario)

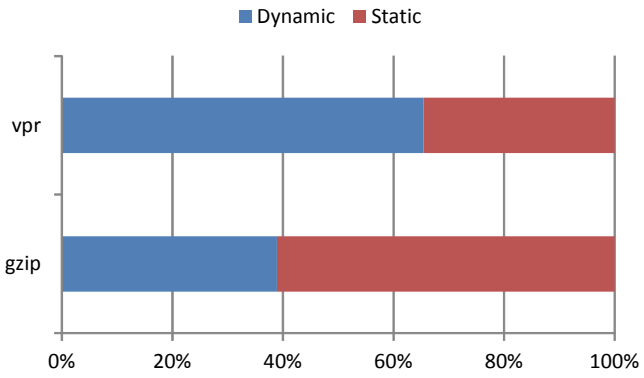

Fig. 5. Breakdown of Power Consumption

Figures 6 and 7 presents relative power consumption in the case where the dynamic power of DDU is twice larger than that of SDU and the ratio of the static power over the total power is 40% and 30%, respectively. As the ratio of the static power over the total power of the baseline becomes smaller, almost all benchmarks are categorized in the second type and the benefit from the proposed technique is reduced. This is as expected because the dynamic power dominates in these cases. Instead, we propose the use of heterogeneous functional units in order to reduce static power consumption.

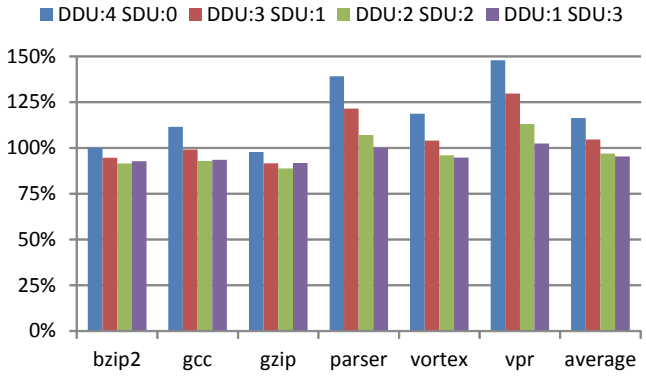

Fig. 6. Relative Power Consumption (40%-scenario)

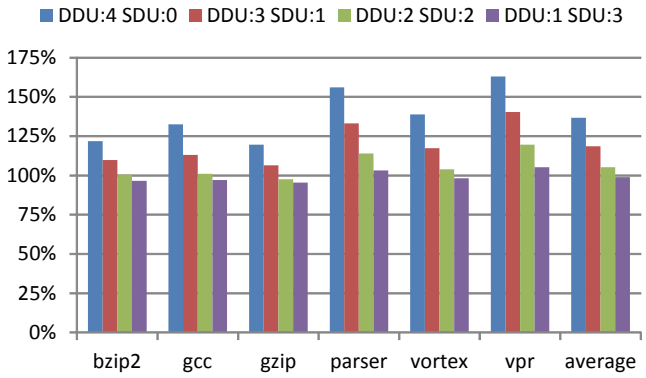

Fig. 7. Relative Power Consumption (30%-scenario)

Figures 8 and 9 present the relative power consumption when we vary the ratio of the dynamic power of DDU over that of SDU between 1 and 2 in the cases of 50% and 30% scenarios, respectively. The horizontal axe indicates the ratio and the vertical axe indicates the relative power consumption. The four lines from top to bottom (see the leftmost) present the cases where 1, 2, 3, and 4 SDU's are replaced by DDU's, respectively. As expected, the smaller the ratio is, the larger the power reduction is.

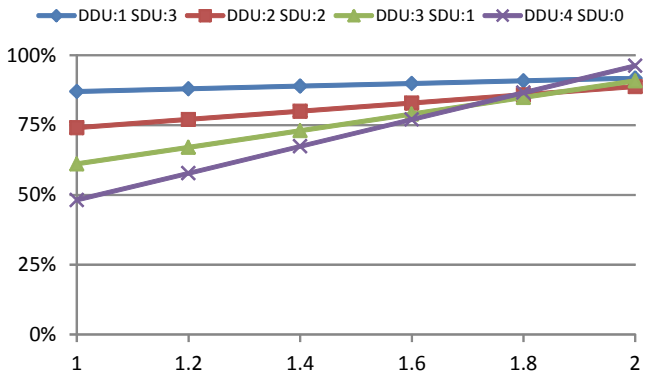

Fig. 8. Sensitivity to Dynamic power of DDU (50%-scenario)

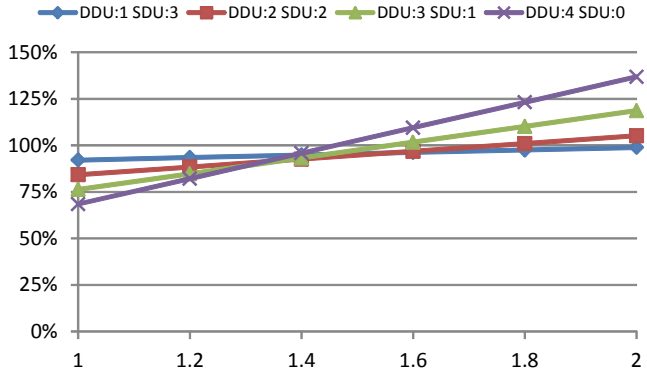

Fig. 9. Sensitivity to Dynamic power of DDU (30%-scenario)

V. Conclusions

This paper proposed to utilize power heterogeneous functional units in order to reduce the total power consumption of microprocessor cores. From the detailed simulations, 11.2% of power reduction is achieved if 2 of 4 leaky functional units are replaced by leakless ones in the situation where the static power occupies half the total power.

Acknowledgment

This work is supported in part by JSPS Grant-in-Aid for Challenging Exploratory Research and by the fund from Central Research Institute of Fukuoka University.

References

[1] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold Leakage Modeling and Reduction Techniques," IEEE/ACM International Conference on Computer-Aided Design, 2002.

[2] D. Ikebuchi, N. Seki, Y. Kojima, M. Kamata, L. Zhao, H. Amano, T. Shirai, S. Koyama, T. Hashida, Y. Umahashi, H. Masuda, K. Usami, S. Takeda, H. Nakamura, M. Namiki, and M. Kondo, "Geyser-1: A MIPS R3000 CPU core with Fine Grain Runtime Power Gating," IEEE Asian Solid-State Circuits Conference, 2009.

[3] A. L. Shimpi, B. Klug, and V. Gowri, "Apple iPad 2 Preview," www.anandtech.com, 3/12, 2011.

[4] NVIDIA Corporation, "Variable SMP (4-PLUS-1™) - a Multi-Core CPU Architecture for Low Power and High Performance," whitepaper, 2011.

[5] ARM Ltd., "The ARM Cortex-A9 Processors," whitepaper, 2009.

[6] M. D. Powell, E. Schuchman, and T. N. Vijaykumar, "Balancing Resource Utilization to Mitigate Power Density in Processor Pipelines," IEEE/ACM International Symposium on Microarchitecture, 2005.

[7] T. Matsumura, T. Ishihara, and H. Yasuura, "Simultaneous Optimization of Memory Configuration and Code Allocation for Low Power Embedded Systems," ACM Great Lakes Symposium on VLSI, 2008.

[8] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," IEEE Computer, Vol. 35, No. 2, February 2002.