Leakage Energy Reduction in Register Renaming

Masaharu Goto

Graduate School of Life Science
and System Engineering
Kyushu Institute of Technology
masaharu@mickey.ai.kyutech.ac.jp

Toshinori Sato

PRESTO
Japan Science and Technology Agency
toshinori.sato@computer.org

Abstract— Register files are becoming a power-hungry component in future embedded microprocessors, as a lot of power reduction techniques are applied, especially on caches, which are currently the most power-hungry component. As higher performance is required for coming smart embedded systems, out-of-order execution, which requires a large number of registers, will be adopted in embedded processors. In addition, deep submicron semiconductor technology leads to larger leakage current. All these situations combine to increase leakage energy consumed by registers in embedded processors. In this paper, we propose a solution for this problem. By exploiting the characteristics of register renaming and the benefit from CMOS circuit techniques with sleep mode, we achieve leakage energy reduction of up to 53.6%.

Keywords— leakage energy, deep submicron, embedded processors, register renaming, superscalar processors

I. Introduction

As many power reduction techniques, especially on caches, have been applied, power consumed by register file becomes a considerable fraction of a total power in embedded microprocessors[20, 21]. For example, in Motorola's M.CORE architecture, the register file consumes 16% of the total processor power and 42% of the datapath power[20, 21]. In addition, out-of-order execution technique has been already adopted in embedded microprocessors[1, 19], and it is announced that simultaneous multi-threading technique will be adopted also in embedded and network processors[4, 5, 7]. Hence, wide-issue out-of-order microprocessors will be widely used in various platforms such as smart broadband devices[19]. In out-of-order processors, a large number of physical registers tend to be implemented for performing register renaming[6, 10, 22]. For example, Intel Pentium 4 has 128 physical registers[10].

On the other hand, leakage energy consumption is becoming more and more important for embedded microprocessor design, as the transistor supply voltage and threshold voltage are scaled down in nanometer-scale technologies.

Considering these trends, embedded microprocessors will employ out-of-order execution and register renaming in the future, resulting in the increase of registers and their power consumption, especially that due to leakage current. From these observations, we have decided to study on reducing leakage energy in register renaming. We utilize CMOS circuits with sleep mode to eliminate energy consumed by unused registers. The rest of this paper is organized as follows: Section II introduces the background on leakage energy. Section III explains how register renaming works in out-of-order processors and presents our concept to reduce leakage energy in register renaming. Section IV presents experimental results and discussion on the effectiveness of our approach. Finally, Section V concludes the paper.

II. Background

A. Leakage Energy

Power consumption in a CMOS digital circuit is governed by the equation:

$$P = P_{active} + P_{off} \quad (1)$$

where P_{active} is the active power and P_{off} the leakage power. The active power P_{active} and gate delay t_{pd} are given by

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

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_t)^\alpha} \quad (3)$$

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 in advanced MOSFETs[11]. Based on Eq.(2), it can easily be found that a power-supply reduction is the most effective way to lower power consumption. However, Eq.(3) 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. 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 leakage power can be given by

$$P_{off} = I_{off} V_{dd} \quad (4)$$

where I_{off} is the leakage current. The subthreshold leakage current I_{off} is dominated by threshold voltage V_t in the following equation:

$$I_{off} \propto 10^{-\frac{V_t}{S}} \quad (5)$$

where S is the subthreshold swing parameter and is around 85mV/decade[18]. Thus, lower threshold voltage leads to increase subthreshold leakage current and static power. Maintaining high transistor switching speeds via low threshold voltage gives rise to a significant amount of leakage power consumption.

B. Leakage Reduction Techniques

Gated-V_{dd}[15] shuts off the supply voltage to SRAM cells to reduce leakage current. The circuit technique has a disadvantage. Gated-V_{dd} loses the state within the memory cell in the sleep mode. If the state should be recovered, there must be an additional dynamic power consumption.

To keep the state, several circuits are proposed such as VT-CMOS[8] and ABB-MTCMOS[12], which reduce leakage current by dynamically raising the threshold voltage by modulating backgate bias voltage. As shown in Table I, Hanson et al.[9] report that leakage current in sleep mode is reduced by a factor of 160 compared with that in normal mode.

TABLE I
Impact of V_t on Leakage Current

Mode	I_{off} (nA)
Sleep	12
Normal	1941

Another approach to reduce leakage current is Drowsy Caches[3], which also keeps the state within the memory cell. It has two different supply voltages and utilizes a dynamic voltage scaling (DVS) technique to reduce static power consumption rather than to trade off dynamic power consumption and performance. Due to short channel effects in deep submicron processes, leakage current is significantly reduced with voltage scaling. When ABB-MTCMOS would be no longer effective, Drowsy Caches is a promising candidates for static energy reduction.

III. Register Renaming

A. Terminology

Several definitions are given here to simplify future references in this section. Modern microprocessors fetch multiple instructions per cycle. Following instruction fetch, the instructions are decoded and issued into instruction window. We use the

term issue to indicate the process of placing the instructions into the instruction window. The instruction window consists of instruction queue and a buffer maintaining program order such as reorder buffer[17], and is operated as a FIFO buffer. The instructions remain in the instruction queue until their operands have been ready. Once their dependences have been resolved, instruction dispatch logic schedules the instructions and then dispatches them into functional units. The instruction queue entries containing the dispatched instructions are deallocated so that new instructions may be issued. We use the term dispatch to move the instructions from the instruction queue to the functional units, where they are executed. After completion of execution, the instructions still wait in the instruction window until their preceding instructions have been retired from the instruction window. When the instructions reach the head of the instruction window, they are retired from it. The instructions may be completed out-of-order but are retired in-order. We use the term instruction queue as the structure holding the instructions waiting for dispatch. On the other hand, the term instruction window is used as the structure holding the issued and completed instructions as well as the waiting ones.

B. Register Renaming Mechanism

In order to eliminate anti- and output- dependences, modern dynamically scheduled processors perform register renaming. There are two common ways to implement the register renaming. One is using a separated renaming registers which are usually constructed by the reorder buffer. The other combines the renaming registers with architected registers in a single register file[6, 10, 22]. We will focus on the latter one.

Figure 1 depicts a register mapping hardware with including its instruction window. The register mapping hardware mainly consists of three structures — map table, reorder buffer, and free list. By means of the map table, each logical register is mapped into a physical register. The destination register is mapped to a free physical register which is supplied by the free list, while operand registers are translated into the last mapping assigned to them. The old destination register is kept in the reorder buffer. When an instruction is retired, the old destination register which is allocated by the previous instruction with the same logical destination register is freed and is placed in the free list. The translated operand registers are held in the instruction queue as tags which are used for Tomasulo's algorithm. Busy bit table contains a bit indicating whether each physical register contains a valid value. It is used for initializing ready bits in the instruction queue for ready operands.

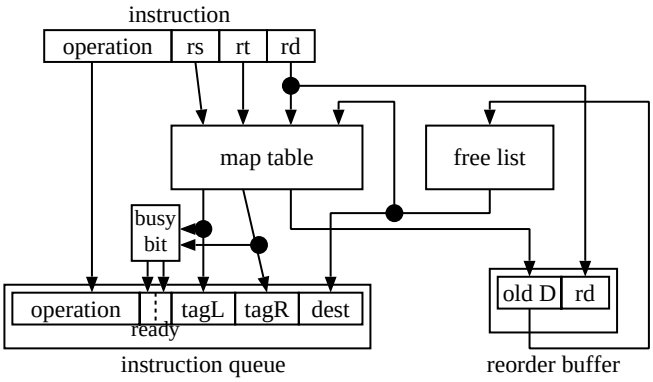

FIG. 1 Register renaming

C. Leakage Energy Reduction in Register Renaming

Figure 2 presents the average number of registers which are active each cycle[1] . We can see that only 26 registers are active. This might imply the unnecessary of a large number of registers. However, it does not. While the average number of active registers are small, large register files are required for keeping high performance. For example, Postiff et al.[14] reported that 64 or more registers enables performance improvement of 20% over the conventional 32 registers. Here, we can find the opportunity for energy reduction. Since there are a lot of registers, which are necessary for high performance but are not always active, we can achieve a significant energy reduction if the registers are turned off while inactive.

[1] Section IV will explain simulation environment for obtaining the numbers.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Next IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	BkCk	Drive

FIG. 3 Pipeline in Intel Pentium 4

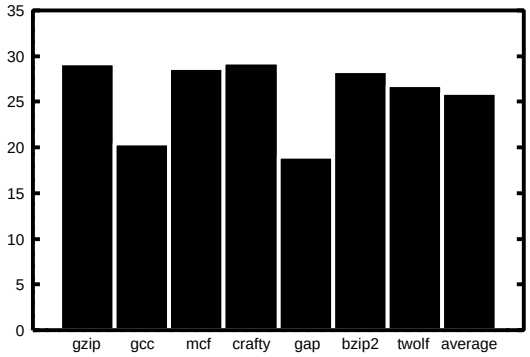

FIG. 2 Average # of active registers

To eliminate leakage energy consumed by inactive registers, we propose to let them in sleep mode. The open issue is how to identify inactive registers. In this study, we adopted a very straightforward technique. As explained above, the free list keeps free physical registers, and hence we can easily identify inactive registers by referring the free list. It is not necessary to refer the free list every cycle. Instead, we should only snoop registers which are both supplied by and placed in the free list. Thus, there is not serious additional power consumption for identifying inactive registers.

On circuits with sleep mode, we can select the most energy-efficient and the least area-overhead one provided by the semiconductor technology in the time of its implementation. This is because the state with in the memory cell can be lost. Values in registers being placed into the free list are no longer used in the future execution of the program. Thus, any one of gated-V_{dd}, VT-CMOS, ABB-MTCMOS, and Drowsy Caches is applicable for this leakage energy reduction technique in register renaming.

When each free register is supplied by the free list, it is turned on into normal (active) mode. There used to be some transition time to turn on transistors[3, 8, 12, 15]. The proposed technique tolerates the transition time. The register only has to be in normal mode before its associated instruction writeback the result into it. There is a considerable period of time after the register is mapped until it is written back. For example, Figure 3 shows a key pipeline in Intel Pentium 4[10]. Each register is supplied in the 7th and 8th stages and is written back after the 20th stage. There is much time for waking up sleep transistors and thus the transition time is hidden.

From these observations, our proposed technique has the least additional power and performance penalties.

D. Related Work

Tseng and Asanovic[21] proposed to reduce dynamic power consumed by a register file in embedded processors. They are circuit-level techniques and thus can be combined with our proposal. Takamura et al.[20] proposed to eliminate register file access by utilizing efficiently operand bypass network, resulting in significant dynamic power reduction. Park et al.[13] also exploited the characteristics of microprocessors that 50 – 70% of operands come from bypass network, and proposed to reduce register ports for power reduction. These previous works focus on dynamic power reduction, and does not consider leakage energy. In contrast, we proposed to reduce leakage energy in register renaming.

IV. Results

We built our evaluation environment using sim-alpha simulator[2], which is a derivative from SimpleScalar tool set[16]. We chose it because HP 21264's microarchitectural implementation on register renaming is modeled in detail in the simulator. Table II shows the configuration of the processor model used in this evaluation. It follows HP 21264's configuration as much as possible.

TABLE II
PROCESSOR CONFIGURATION

fetch width	4 instructions
issue width	4 instructions
dispatch width	4 int + 2 fp instructions
commit width	11 instructions
insn. queue size	20 int + 15 fp instructions
reorder buffer size	80 instructions
load queue size	32 instructions
store queue size	32 instructions
# of func. units	iALU 4, iMUL 4, fALU 1, fMUL/DIV 1
# of physical reg.s	int 41, fp 41
branch predictors	tournament predictor, 8b-history 1K-entry local, 4b-history 4K-entry ghare, 4b-history 4K-entry choice, 32-entry return stack
L1 insn. cache	64K, 2-way set-associative, 64-bytes blocks
L1 data cache	64K, 2-way set-associative, 64-bytes blocks, 8 MSHRs
L2 shared cache	2M, 2-way set-associative, 64-bytes blocks
instruction TLB	128-entry, full-associative
data TLB	128-entry, full-associative

The SPEC2000 CINT benchmark suite is used for this study. We use the object files provided by University of Michigan[16].

Figure 4 shows the percent reduction of leakage power consumed by integer register file. Since we used only integer application programs, we do not consider floating-point register file. If we include power reduction in floating-point register file, the percent number is larger than that shown in Figure 4. As explained in Section III, there are no increase in execution cycles, and thus power reduction directly turns into energy reduction. As we can see, an average energy reduction of 36.7% is achieved. In the cases of `gcc` and `gap`, more than half of leakage energy consumed in integer register file is eliminated.

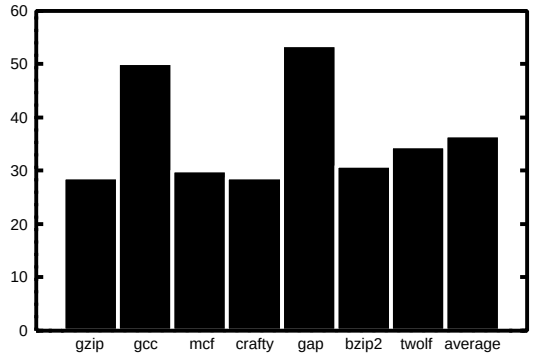

FIG. 4 % REDUCTION IN LEAKAGE POWER

V. CONCLUSIONS

Register files are becoming one of the most power-hungry component in embedded microprocessors, as high performance is required for smart embedded devices and as semiconductor technology is improved. This paper proposed to reduce leakage energy consumed in register files using CMOS circuit with sleep mode. We exploit the beneficial characteristics of register renaming that inactive registers are easily identified and transition time of the circuits is hidden. From the detailed microarchitectural-level simulation, we found that as much as 53.6% and an average of 36.7% of leakage energy consumed in integer register file is eliminated.

REFERENCES

[1] M. Daito, K. Suzuki, K. Uehigashi, H. Morita, H. Sonoda, N. Morikawa, M. Moriyama, S. Sato, T. Fukuda, and S. Nakamura, "A 500-MHz Embedded Out-Of-Order Superscalar Microprocessor," IEICE Transactions on Electronics, Vol.E85-C No.2, 2002.

[2] R. Desikan, D. C. Burger, and S. W. Keckler, "Measuring Experimental Error in Microprocessor Simulation," International Symposium on Computer Architecture, 2001.

[3] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy Caches: Techniques for Reducing Leakage Power," International Symposium on Computer Architecture, 2002.

[4] P.N.Glaskowsky, "Networking Gets XStream," Microprocessor Report, Vol.14, Archive 11, 2000.

[5] P. N. Glaskowsky, "MemoryLogix Makes Tiny x86," Microprocessor Report, Vol.16, Archive 11, 2002.

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

[7] J. C. Kim, "The Halla: An ARM10 Compatible 1.2GHz Processor," NE Embedded Processor Symposium, 2002.

[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] H. Hanson, M. S. Hrishikesh, V. Agarwal, S. W. Keckler, and B. Burger, "Static Energy Reduction Techniques for Microprocessor Caches," IEEE Transactions on VLSI Systems, to appear.

[10] 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.

[11] 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.

[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] Il Park, Michael Powell, and T. N. Vijaykumar , "Reducing Register Ports for Higher Speed and Lower Power," International Symposium on Microarchitecture, 2002.

[14] M. Postiff, D. Greene, and T. Mudge, "The Need for Large Register Files in Integer Codes," Technical Report CSE-TR-434-00, Department of Electrical Engineering and Computer Science, University of Michigan, 2000.

[15] M. Powell, S. H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," International Symposium on Low Power Electronics and Design, 2000.

[16] SimpleScalar LLC, `www.simplescalar.com`.

[17] J. E. Smith and A. R. Pleszkun, "Implementation of Precise Interrupts in Pipelined Processors," International Symposium on Computer Architecture, 1985.

[18] D. Sylvester and H. Kaul, "Power-Driven Challenges in Nanometer Design," IEEE Design & Test of Computers, Vol.18, No.6, 2001.

[19] K. Tajima, "VR7700: A High Performance Out-Of-Order Superscalar Microprocessor with Integrated L2 Cache and Peripherals," COOL Chips VI, 2003.

[20] H. Takamura, K. Inoue, and V. Moshnyaga, "Register File Access Reduction by Data Reuse," International Workshop on Power And Timing Modeling, Optimization and Simulation, 2002.

[21] J. Tseng and K. Asanovic, "Energy-Efficient Register Access" Symposium on Integrated Circuits and System Design, September 2000.

[22] K. C. Yeager, "The MIPS R10000 Superscalar Microprocessor," IEEE Micro, Vol.16, No.2, 1996.