A Static and Dynamic Energy Reduction Technique for I-Cache and BTB in Embedded Processors

Hidenori Sato[1] Toshinori Sato[1,2]

[1] Department of Artificial Intelligence, Kyushu Institute of Technology
[2] PRESTO, Japan Science and Technology Agency
E-mail: {hide,tsato}@mickey.ai.kyutech.ac.jp

Abstract— Power consumption is becoming one of the most important constraints for embedded processor design in nano-meter-scale technologies. Especially, as the transistor supply voltage and threshold voltage are scaled down, leakage energy consumption is increased even when the transistor is not switching. This paper proposes to use the loop cache to reduce static energy consumption as well as dynamic one. We combine it with CMOS circuits having sleep mode, and thus instruction cache can go to sleep mode when the loop cache is active. Detailed simulation shows that we can reduce static energy consumed by I-cache by up to 37.9%. We also propose to apply the technique to branch target buffer, and its static and dynamic energy consumption is reduced by up to 40.4% and 40.7%, respectively.

1 Introduction

Power consumption is becoming one of the most important concerns for embedded processor designers in nanometer-scale technologies. Until recently, the primary source of energy consumption in digital CMOS circuits has been the dynamic power that is caused by dynamic switching of load capacitors. The trend of the reduction in transistor size reduces capacitance, resulting in less dynamic power consumption. Microprocessor designers have relied on scaling down the supply voltage, resulting in further dynamic power reduction [8, 9, 13]. In addition, many architectural technologies to reduce power have been proposed by reducing the number of switching activities [11, 15, 16]. To maintain performance scaling, however, threshold voltage must also be scaled down with supply voltage. Unfortunately, this increases leakage current exponentially. The International Technology Roadmap for Semiconductors (ITRS) predicts an increase in leakage current by a factor of two per generation [19]. Borkar estimates a factor of 5 increases in leakage energy in every generation [1].

Many techniques [4,10,12,18] proposed to address this problem have focused on cache memory that is a major energy consumer of the entire system because leakage energy is a function of the number of transistors. For example, the StrongARM processor uses 60% of the die area for cache memories [16]. Current efforts on static energy reduction do not consider dynamic energy reduction [4,10,12,18]. In this paper, we propose a technique to reduce static and dynamic energy consumption simultaneously.

2 Motivations

2.1 Leakage Energy

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

$$P = P_{active} + P_{off} \tag{1}$$

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

$$P_{active} \propto f C_{load} V_{dd}^2 \tag{2}$$

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

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 [7]. 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

$$P_{off} = I_{off} V_{dd} \tag{4}$$

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

$$I_{off} \propto 10^{-\frac{V_t}{S}} \tag{5}$$

where S is the subthreshold swing parameter and is around 85mV/ decade [19]. 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.

2.2 Background on Leakage Energy Reduction

There are several circuits proposed to reduce leakage current by dynamically raising the threshold voltage, for example by modulating backgate bias voltage [14, 17]. Figure 1 shows ABB-MTCMOS circuit [17]. The four transistors with high threshold voltage (denoted as `H-Vt`) become the switch that cut off leakage current. During normal operation, when sleep signal `SL` is deasserted, the virtual source line, `Vvdd`, and the back gate bias line, `BP`, are set to appropriate power supply, `Vdd1`. During sleep mode, when `SL` is asserted, P-well is biased using the alternative power supply, `Vdd2`, at a higher voltage level. Diodes perform backgate bias effect and thus threshold voltage of all transistors is increased. This reduces leakage current, which flows from `Vdd2` to the ground. As shown in Table 1, Hanson et al. [6] report that leakage current in sleep mode is reduced by a factor of 160 compared with that in normal mode. VT-CMOS [14] also controls the backgate bias to

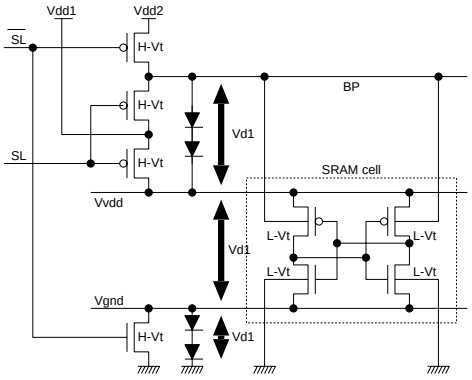


Figure 1: ABB-MTCMOS Circuit

Table 1: Impact of V_t on Leakage Current

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

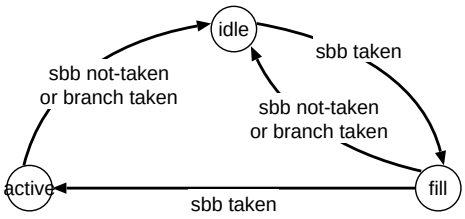


Figure 2: Loop Cache Controller

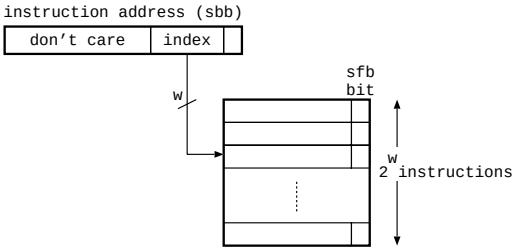


Figure 3: Extended Loop Cache

reduce leakage current in sleep mode by rising up threshold voltage.

Another approach to reduce leakage current is Drowsy Caches [3]. 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.

2.3 Related Works

Current efforts on static energy reduction have focused on dynamically resizing active area of caches [4, 10, 12, 18]. These architectural techniques employ circuit techniques. VT+ADR cache [4] and SA cache [10] use VT-CMOS [14] and ABB-MTCMOS [17], respectively. Decay cache [12] and DRI cache [18] use gated-V_{dd} [18], which 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. Thus, additional cache messes might occur, resulting in an additional dynamic power consumption. In contrast, VT-CMOS and ABB-MTCMOS can retain stored data in the sleep mode. However, these architectural techniques focus on static energy reduction and dynamic energy consumption is not considered.

3 Energy Reduction via Loop Cache

The Loop cache [15] is a small instruction buffer proposed for dynamic power reduction. We use the loop cache to reduce static energy consumption as well as dynamic one. In addition, we will apply the technique explained below also to BTB.

3.1 Loop Cache

The Loop cache [15] is a small instruction buffer proposed for dynamic power reduction. It is placed in parallel with L1 I-cache instead of placed between I-cache and CPU, and instructions are supplied to CPU either from the loop cache and I-cache. The loop cache controller knows precisely whether the next instruction request will hit in the loop cache, based on detecting a tight loop, which defined as any short backward branch instruction (sbb). At the end of the first iteration of a tight loop, the sbb is detected. During the second iteration, the loop cache is filled from I-cache. And last, after the third iteration, instructions are supplied from the loop cache instead of I-cache.

The state machine for the loop cache controller is shown in Figure 2 [15]. There are three states: idle, fill, and active. When a sbb is detected during the idle state, and if the sbb is taken, the controller enters the fill state and begins to fill instructions into the loop cache. During the fill state, if any branch instruction except for the sbb is taken, the controller returns to the idle state. In the second iteration, if the sbb is taken again, the controller enters the active state. Otherwise, it returns to the idle state. In the active state, instructions are supplied by the loop cache, and I-cache is not accessed. Because dynamic power consumed by the loop cache is much smaller than that consumed by I-cache, considerable power reduction is achieved. During the active state, if the sbb is not-taken, or if any other branch is taken, the controller returns to the idle state.

3.2 Extension to Loop Cache

One drawback of the loop cache is that any loop including taken branch other than a sbb can not utilize the loop cache. This diminishes the loop cache utilization. For example, `IF-THEN-ELSE` statement emerges frequently in the cases where error checking is required. In order to solve this problem, we propose not to leave the fill state when a taken branch other than the sbb is a short forward branch (sfb). If the target of the sfb is between the sfb and the sbb, the loop cache controller stays in the fill state.

Figure 3 shows the extended loop cache. Each entry of the loop cache has an additional sfb bit. When an instruction is a taken sfb, its corresponding sfb bit is set. During the active state, the loop cache controller checks the sfb bit whenever any branch other than the sbb emerges. Even when the branch is taken, the controller doesn't leave the active state if its corresponding sfb bit is set. It should be noted that even if the branch is not taken, the controller has to return to the idle state if its corresponding sfb is reset. This extension to the loop cache will improve its utilization.

3.3 Static Energy Reduction via Loop Cache

In order to reduce leakage energy consumed in I-cache, we proposed to combine the loop cache with CMOS circuits having sleep mode, such as ABB-MTCMOS and Drowsy Caches explained in Section 2. Even when instructions are supplied from the loop cache, static power is consumed by I-cache. Therefore, we propose to move I-cache into sleep mode, where threshold voltage of all transistors is increased and thus leakage current is reduced, during the loop cache is active. Because I-cache is not accessed during the active mode, almost no latency penalty is expected. Only when the loop cache controller returns to the idle mode from the active mode, a penalty is required to activate the sleeping I-cache. Because a tight loop stored in the loop cache has a lot of iterations, this penalty is expected to be negligible.

3.4 Reducing Energy Consumed by BTB

Because BTB has a similar structure with caches and it is accessed every cycle, it consumes much energy. Thus, it is expected to reduce energy consumed by BTB. To attack the problem, we propose to apply the technique for I-cache explained above to BTB. During CPU executes a tight loop stored in the loop cache, the sbb is expected to be taken with high probability. In other words, every sbb is always predicted taken. In addition, it is not required to predict target instruction of any branch instruction other than the sbb, because its next instruction is stored in the next entry in the loop cache. Thus, during the loop cache is active, BTB is not required. From these observations, we propose to stop accessing BTB and to move BTB into sleep mode during the loop cache is active. Using the technique, both dynamic and static energy consumed by BTB is reduced. Only when the loop cache controller returns to the idle mode from the active mode, CPU suffers a missprediction penalty. Because a tight loop stored in the loop cache has a lot of iterations, this penalty is expected to be negligible.

4 Experimental Results

4.1 Simulation Methodology

We implemented our simulator using the SimpleScalar/ARM tool set [2]. We use ARM instruction set architecture in this study. The processor model evaluated is based on Intel XScale processors [8]. 32KB, 32B block, 32-way set-associative L1 I-cache and data cache (D-cache) are used. They have a load latency of 1 cycle and a miss latency of 32 cycles. When in sleep mode, L1 I-cache has a latency of 2 cycles, 1 cycle of which is required to activate the sleeping I-cache [3, 6, 10] and another cycle of which is for reading an instruction. The replacement policy is based on LRU. No L2 cache is used. A memory operation that follows a store whose data address is unknown cannot be executed.

The MiBench [5] is used for this study. It is developed for use in the context of embedded, multimedia, and communications applications. It contains image processing, communications, and DSP applications. We use original input files provided by University of Michigan. Table 2 lists the benchmarks we used. All programs are compiled by the GNU GCC with the optimization options specified by University of Michigan. Each program is executed to completion.

4.2 Results

Figure 4 shows the percentage that instruction request hits in the loop cache. The vertical line indicates the access rate and the horizontal line indicates the loop cache size. For each group of four bars, the first one (denoted as `base`) indicates the access rate for the base loop cache model. The second one (denoted as `+sfb`) is for the extended loop cache model where the loop cache can hold some short forward branch instructions. The third one (denoted as `-btb`) is for the base loop cache model where BTB is turned off when the loop cache is active. And the last one (denoted as `+sfb-btb`) is for the extended loop cache model with the BTB turn-off strategy.

Table 2: Benchmark programs

program	input set
Rawdaudio	ADPCM decode
CRC	32-bit Cyclic Redundancy Check
FFT	Fast Fourier Transform
Toast	GSM encode

Table 3: %Energy Reduction (with 64-entry Loop$)

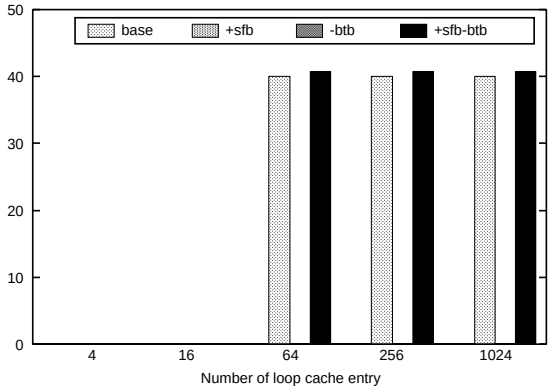
program	static	dynamic
	I-cache	BTB
Rawdaudio	37.9	40.4	40.7
CRC	-0.04	0.0	0.0
FFT	14.2	15.2	15.3
Toast	19.4	20.8	20.9

First, in the case of `CRC`, the loop cache is never used for all cache sizes we evaluated. This means that the loop cache does not contribute to energy reduction. In this case, energy consumption must be increased due to dynamic power consumed by the loop cache controller and static power consumed by the loop cache. Second, it can be seen that the extension to the loop cache improves the loop cache utilization, especially for `rawdaudio`. If this extension is not applied, the loop cache is not useful for `rawdaudio` as well as `CRC`. This contributes to dynamic and static power reduction in I-cache. Only in the case of `FFT` with the 1024-entry loop cache, the extension reduces the loop cache utilization. However, the loss is less than 3%. Third, even if BTB is turned off during the loop cache is active, there is no difference in the loop cache access rates. Thus, this results in power reduction in BTB. And last, it is found that combining the two techniques evaluated individually above has no sever impact on the loop cache access rates.

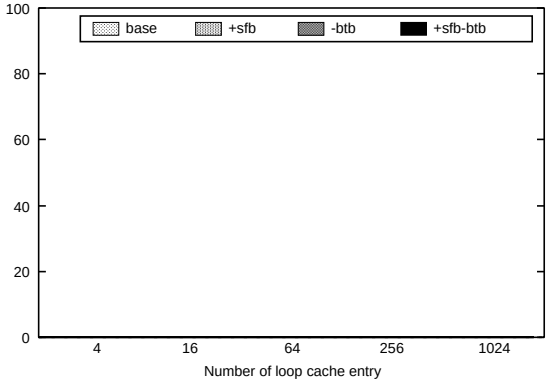
As we have seen, the proposed techniques have the potential to reduce static and dynamic power consumed in I-cache and BTB. However, if execution cycles were increased, the total energy would not be reduced. From the timing simulations, we can find that execution cycles is increased by less than 0.1% for all programs. Therefore, it is confirmed that our proposals can improve energy efficiency by reducing static and dynamic power consumed by I-cache and BTB and by maintaining execution cycles. Table 3 presents energy reduction when the 64-entry extended loop cache with BTB turn-off strategy is used. It can be seen that static energy consumed by I-cache is reduced by up to 37.9% and that consumed by BTB is reduced by up to 40.4%. Only in the case of `CRC`, I-cache static energy is increased by 0.04%. It is also found that dynamic energy consumed by BTB is reduced by up to 40.7%.

5 Conclusions

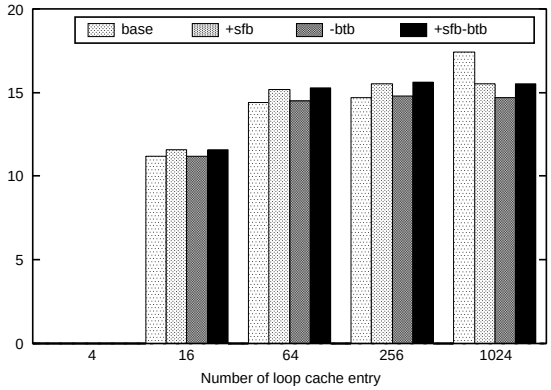
This paper proposed to use the loop cache to reduce static and dynamic energy consumed in I-cache and BTB. We combined it with CMOS circuits having sleep mode, and thus I-cache and BTB can go to sleep mode when the loop cache is active. Using the 64-entry loop cache, up to 40% I-cache and BTB accesses are eliminated, resulting in dynamic power reduction in these blocks. In addition, because they are in sleep mode during the loop cache is in active mode, static power is also reduced. Even if I-cache and BTB are turned off, execution cycles are



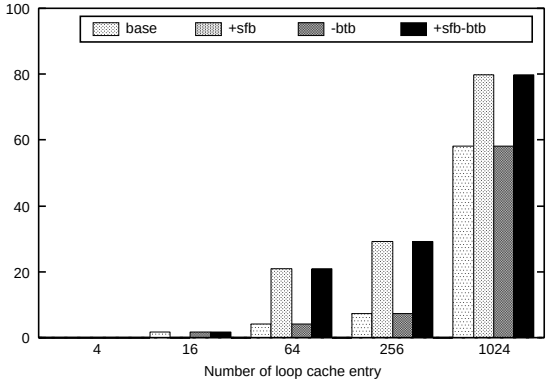
(i) Rawdaudio



(ii) CRC



(iii) FFT



(iv) Toast

Figure 4: %Loop cache access rates

maintained, resulting in energy consumption reduction.

References

[1] S. Borker, "Design challenges of technology scaling," IEEE Micro, vol.19, no.4, 1999.

[2] D. Burger, T. M. Austin, "The SimpleScalar tool set, version 2.0," ACM SIGARCH Computer Architecture News, vol.25, no.3, 1997.

[3] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, T, Mudge , "Drowsy caches: techniques for reducing leakage power," Int. Symp. on Computer Architecture, 2002.

[4] R. Fujioka, K. Katayama, R. Kobayashi, H. Ando, T. Shimada, "A preactivating mechanism for a VT-CMOS cache using address prediction," Int. Symp. on Low Power Electronics and Design, 2002.

[5] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," Workshop on Workload Characterization, 2001.

[6] H. Hanson, M. S. Hrishikesh, V. Agarwal, S. W. Keckler, and B. Burger, "Static energy reduction techniques for microprocessor caches," IEEE Trans. on VLSI Systems, to appear.

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

[8] Intel Corp., "Intel XScale technology," `http://developer.intel.com/ design/intelxscale/`, 2002.

[9] T. Ishihara, K. Asada, "A system level memory power optimization technique using multiple supply and threshold voltages," Asia and South Pacific Design Automation Conf., 2001.

[10] T. Ishihara, K. Asada, "An architectural level energy reduction technique for deep-submicron cache memories," Asia and South Pacific Design Automation Conf., 2002.

[11] A. Iyer et al., "Power aware microarchitecture resource scaling," Design, Automation and Test in Europe Conf. and Exhi., 2001.

[12] S. Kaxiras, Z. Hu, G. Narlikar, R. McLellan, "Cache-line decay: a mechanism to reduce cache leakage power," Workshop on Power Aware Computer Systems, 2000.

[13] A. Klaiber, "The technology behind Crusoe processors," Transmeta Corporation, White Paper, 2000.

[14] T. Kuroda et al., "A 0.9V, 150MHz, 10mW, 4mm^2, 2-D discrete cosine transform core processor with variable-threshold-voltage scheme," Int. Solid State Circuit Conf., 1996.

[15] L.H,Lee, B.Moyer, J.Arends, "Instruction fetch energy reduction using loop caches foe embedded applications with small tight loops," Int. Symp. Low Power Electronics and Design, 1999.

[16] S. Manne, A. Klauser, D. Grunwald, "Pipeline gating: speculation control for energy reduction," Int. Symp. on Computer Architecture, 1998.

[17] K. Nii, H. Makino, Y. Tujihashi, C. Morishima, Y. Hayakawa, "A low power SRAM using auto-backgate-controlled MT-CMOS," Int. Symp. on Low Power Electronics and Design, 1998.

[18] M. Powell, S. H. Yang, B. Falsafi, K. Roy, T. N. Vijaykumar, "Gated-Vdd: a circuit technique to reduce leakage in deep-submicron cache memories," Int. Symp. on Low Power Electronics and Design, 2000.

[19] D. Sylvester, H. Kaul, "Power-driven challenges in nanometer design," IEEE Design & Test of Computers, vol.18, no.6, 2001.