A Preactivating Mechanism for a VT-CMOS Cache using Address Prediction

Ryo Fujioka, Kiyokazu Katayama, Ryotaro Kobayashi, Hideki Ando, Toshio Shimada
Department of Information Electronics, Graduate School of Engineering, Nagoya University
Furo, Chikusa, Nagoya, Aichi
464-8603 Japan

{fujioka,katayama,kobayasi,ando,shimada}@shimada.nuee.nagoya-u.ac.jp

ABSTRACT

It has become an important requirement to achieve high performance and low-power consumption at the same time. The dynamic leakage cut-off (DLC) scheme, which controls transistors' threshold voltage by the line on demand, is a technique that potentially satisfies that requirement for a cache. Yet, conventional DLC causes access time to significantly lengthen, and consequently processor performance is unacceptably degraded. This paper proposes a mechanism that suppresses the performance degradation by preactivating cache lines using address prediction before access requests. Our evaluation results show significant performance improvements are achieved with little increase of power consumption.

Keywords

leakage current, L1 data cache, address prediction

1. INTRODUCTION

Power consumption is rapidly becoming a critical concern in designing processors, in addition to concerns about performance. An emerging problem as technology advances is the growth of static power consumption and the most important source of static power is subthreshold current.

To solve this problem, several circuit-level solutions have been proposed. *Variable-threshold CMOS* (VT-CMOS) [4] and *multi-threshold CMOS* (MT-CMOS) [6] are two simple but effective circuit techniques. Although these techniques can dramatically reduce the current leakage, they have a common flaw in that the transition time required to activate from standby is very long. Thus, these techniques are only available to infrequently-used blocks where a long activation time has little affect on processor performance. However, this obviously limits any significant power reduction of a processor. We need to carefully control power of frequently-used blocks with little adverse effect on performance.

ISLPED'02, August 12-14, 2002, Monterey, California, USA.
Copyright 2002 ACM 1-58113-475-4/02/0008 ...$5.00.

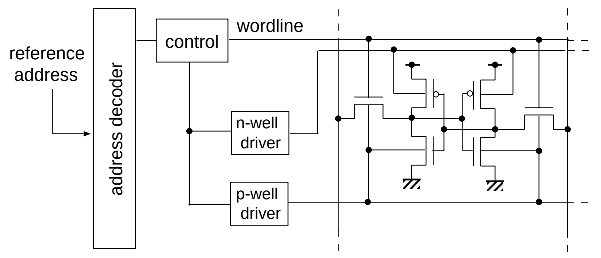

Figure 1: Circuit of a line in the original DLC cache.

Among frequently-used blocks, a cache contains the largest amount of transistors in a processor chip. Thus, it would be effective if the static power of a cache can be reduced. For this purpose, a scheme called the *dynamic leakage cut-off* (DLC) scheme was proposed [3]. In DLC, an SRAM cell is composed of VT-CMOS and power is controlled by a cache line. DLC activates only a selected cache line after address decoding. Although DLC significantly decreases activation time through per-line control, activation still takes long time. Consequently, the access time of a cache unacceptably lengthens.

In this paper, we propose a mechanism, we call *preactivating*, which suppresses performance degradation at the architectural level when DLC is applied to a cache. Specifically, our mechanism predicts the cache line which will be accessed, and the predicted line is activated before the line is actually accessed.

This paper is organized as follows. Section 2 describes our mechanism. Section 3 shows results, and Section 4 presents related work. Finally, Section 5 presents conclusions.

2. PREACTIVATING IN A DLC CACHE

In this section, we detail the original DLC cache and then we propose our mechanism.

2.1 DLC Cache

Figure 1 shows the circuit of a line in the original DLC cache [3]. In DLC, each SRAM cell is composed of VT-CMOS. Any SRAM cells of any unselected lines are initially deactivated. In other words, the threshold voltages of transistors in a cell are high to suppress leakage current. When a reference address is applied and a line is selected after the address is decoded, all cells in the selected line are activated by changing the threshold voltages of transistors to a low

Table 1: Parameters for 0.18μm process.

Transistor						Mid-Level Metal			Top-Level Metal		
Gate Cap. (fF/μm)	pwell-n$^+$ Cap. (fF/μm)	pwell-nwell Cap. (fF/μm^2) Area	pwell-nwell Cap. (fF/μm) Line	Transistor Res. (KΩ-μm)	Supply Voltage (V)	Width (nm)	Res. (mΩ/μm)	Cap. (fF/μm)	Width (nm)	Res. (mΩ/μm)	Cap. (fF/μm)
1.56	1.51	1.00	0.10	3.6	1.8	320	107	0.253	530	36	0.270

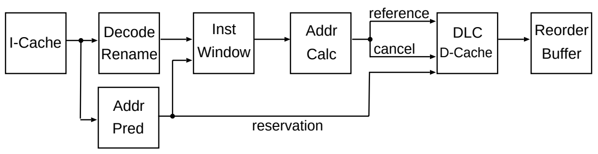

Figure 2: Processor organization with a preactivating DLC cache.

Table 2: Parameters for a memory cell.

transistor width	6λ
p-well size per cell	area of a cell $\times$ 4/6
height of a cell	40λ
width of a cell	20λ

Table 3: Processor parameters.

decode/issue width	8
instruction window	128-entry RUU, 64-entry LSQ
I-cache	perfect, 1-cycle hit latency
D-cache	32KB, 2-way set-associative, 32B line, 4 ports, 1-cycle hit latency, 6-cycle miss penalty

voltage. This is done by the n-well and p-well drivers. Soon after being activated, the word line of the selected line rises. Since the threshold voltages of the accessed cells are low, the delay of the bit line is as short as in a normal cache. Although the activated line consumes significant static power, it is negligibly small in a large cache. Note that the activation is triggered after the address is decoded and the activation time is significantly long (it is estimated in Section 3.1). As a result, the access time becomes considerably longer than a normal cache.

2.2 Mechanism to Reduce Performance Degradation

Figure 2 shows our proposed processor organization (note that the figure does not present a precise pipeline organization; it depends on implementation). Although the organization is basically similar to the usual superscalar processor, it is differentiated by having an address predictor and a D-cache with three address inputs (*reference*, *cancel*, and *reservation*) per memory instruction. The address predictor predicts the address of a location which a memory instruction will access before the address is calculated. Our mechanism uses a *stride value predictor* to predict addresses because it gives the best cost/performance as an address predictor[8, 9].

Figure 3 shows the organization of a line of our DLC cache. The differences from a conventional DLC cache are the two extra address decoders per memory instruction, and an up/down counter we call the *reservation counter* per cache line. The counter indicates the number of reservations for line accesses.

As shown in Figure 2, while instructions are decoded, the addresses of locations which will be accessed by decoding instructions are predicted using the address predictor. The predicted addresses are written into an entry for the corresponding instruction in the instruction window. At the same time, they are sent to the D-cache as reservation addresses, and the reservation counter in each line is increased by one. If the counter value changes from zero to one, the corresponding line is activated to prepare for later actual references. When the counter value is non-zero, the line is continuously activated.

At some cycles later, a memory instruction is issued from the instruction window with the prior-written predicted address. An effective address is then calculated. The effective address is sent to the D-cache as a reference, and the predicted address as a cancel addresses. The reference address is used to access D-cache data as in a usual cache. If the predicted address is correct, the selected cache line is already activated. Thus, delay due to DLC is not incurred; if not correct, a delay may be incurred. Meanwhile, the reservation counter in the selected line accessed by the cancel address is decreased by one. If the counter value becomes zero, the corresponding line is deactivated. Note that when a branch is found to be mispredicted, all reservation counters are reset.

3. EVALUATION RESULTS

In this section, we first estimate the access time penalty incurred by DLC. We then evaluate processor performance and power consumption.

3.1 Estimation of DLC Cache Delay

We estimate how much the activation time impacts on cache access time. In general, delay estimation is not difficult if SPICE is used, but unfortunately we do not have SPICE parameters, which are usually unavailable to the public. Instead, we have used a simulator based on *cacti* [7, 10]. Cacti is a program to calculate cache access time. It analytically models a cache by decomposing the circuit into basic gates and many equivalent RC circuits. Gate delay is estimated based on Horowitz's model [2]. We modified the original cacti so that it could adapt to a 0.18μm current process. The assumed parameters are shown in Table 1. Table 2 also shows the parameters of the memory cell we assumed.

For a given cache organization parameters (capacity, line size, and associativity), we first calculated access time and word line delay using cacti. We then calculated the capacitance ratio of the p-wells to the word line per line. Finally, assuming gate delay proportionally increases to its load capacitance, we estimated the activation time as the product of the word line delay and the capacitance ratio.

Figure 4 shows the evaluation results of the access time of a DLC and a normal non-DLC cache for various capacities. Associativity for both caches are two (associativity has only

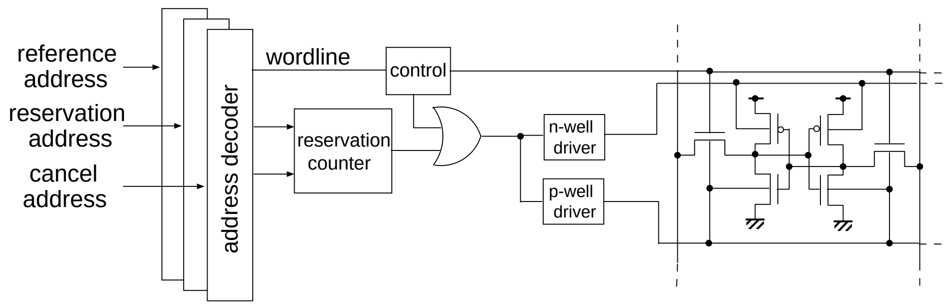

Figure 3: Circuit of a line in the preactivating DLC cache.

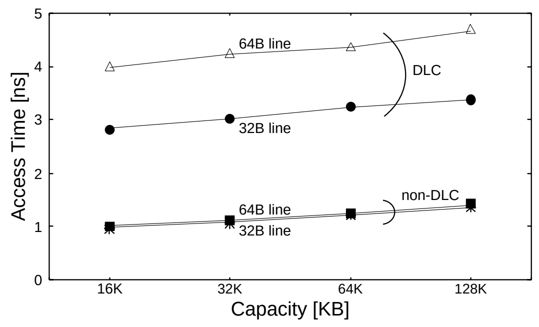

Figure 4: Accesses time of DLC and non-DLC caches.

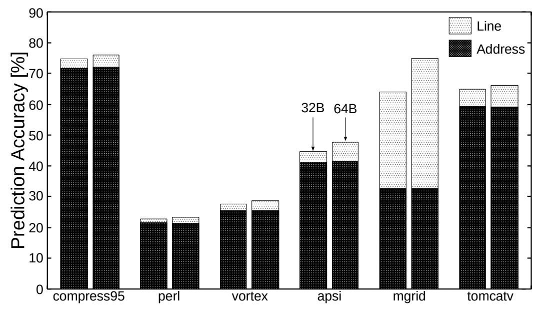

Figure 5: Address and line prediction accuracy.

a small impact according to our simulation, except for direct map). Two lines for each cache present the cases of 32B and 64B lines. As shown in Figure 4, the access time of a DLC cache is much longer than that of the non-DLC cache. As a line size becomes larger, the access time is longer. For example, the 32KB DLC cache with 32B or 64B lines has a 2.7 or 3.9 times longer access time, respectively, than the non-DLC cache.

3.2 Processor Performance

First, we explain our evaluation environment, and then evaluate address prediction accuracy, and the effectiveness of our preactivating mechanism. Finally, we evaluate power consumption.

3.2.1 Evaluation Environment

We evaluated processor performance with SimpleScalar Tool Set [1] version 3.0b. Its instruction set architecture, SimpleScalar/PISA, is extended from MIPS R10000 [5]. We used six benchmark programs from SPEC95. Each benchmark program was compiled by GNU GCC version 2.7.2.3 with -O6 and -funroll-loops optimization flags.

The parameters of a baseline processor are shown in Table 3. The baseline processor has the normal non-DLC D-cache. We compared two processor models: a processor with the conventional DLC cache and that with preactivating DLC cache. In each model, the hit latency of the DLC cache is three or four cycles for the 32B or 64B line cases, respectively, as derived from our evaluation in Section 3.1. However, if the address is predicted correctly, no delay due to DLC is incurred in the preactivating model(described in Section 2.2).

3.2.2 Address and Line Prediction Accuracy

We used a stride predictor with a 1K-entry value history table in our evaluation. Figure 5 shows the evaluation results of address and line prediction accuracy. The left bar of each program presents prediction accuracy for the 32B line case, and the right bar for the 64B line case. Prediction accuracy is the rate of the number of correct predictions to total memory instructions. Line prediction is correct if a predicted address is within an accessed line, independently of the correctness of the address prediction, and thus line prediction is always better than address prediction. The lower portion of each bar represents address prediction accuracy, and the upper portion represents the increase in accuracy by line prediction. Since preactivating is successful if line prediction is correct, line prediction accuracy is more important.

As shown in Figure 5, the line prediction accuracy is 49.8% on average for a line size of 32B. However, accuracy is highly dependent on programs because of their memory access patterns. Good accuracy is achieved in compress95, mgrid, and tomcatv, while it is poor in perl and vortex. Line prediction accuracy increases little from that of address prediction accuracy in most programs, but significantly increases in mgrid. Line prediction accuracy is 52.8% on average for a line size of 64B; little benefit is obtained from a longer line except for with mgrid.

3.2.3 Performance Evaluation

Figure 6 shows performance degradation from a baseline due to DLC caches. The lower portion of each bar represents degradation when preactivating DLC, and the upper portion represents the increase in degradation of a conventional DLC.

As shown in Figure 6, the conventional DLC cache causes

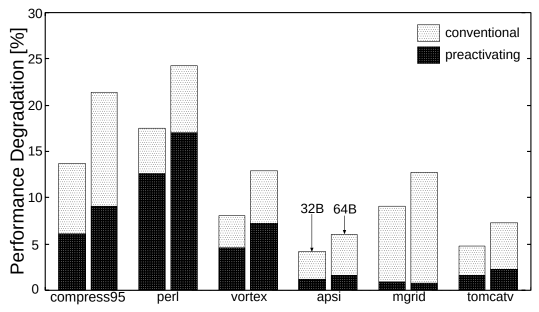

Figure 6: Performance degradation from the baseline due to DLC caches.

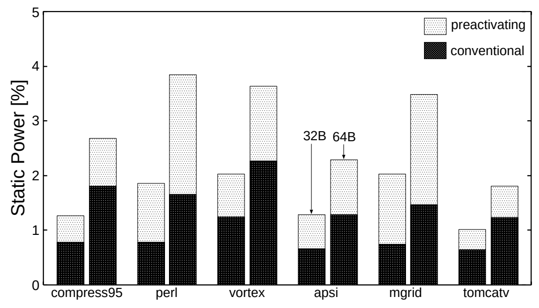

Figure 7: Static power rate to the base line.

a large performance degradation. It is a maximum of 17.5%, average 9.6%, for the 32B line case, and a maximum of 24.3%, average 14.1%, for the 64B line case. The adverse effect on performance due to a long latency of DLC is larger in integer than in floating-point programs. This difference arises from the amount of parallelism contained in a program; in general a floating-point program contains more parallelism than an integer program. An instruction scheduler can hide a long latency by exploiting parallelism, and thus can reduce adverse effects on performance.

As we expected, the preactivating mechanism can alleviate performance degradation more in a program where line prediction is better. The largest reduction of performance degradation is observed in mgrid; the reduction rate is approximately 90% for both line sizes. This is much larger than those of compress95 and tomcatv whose prediction accuracies are as high as mgrid's. The reason is the high frequency of loads in a program; it is 33% in mgrid while 21% in both compress95 and tomcatv. On average, our preactivating mechanism can reduce performance degradation by 47.9% and 55.3% for 32B and 64B lines, respectively. As a result, our preactivation DLC improves performance by 3.2-8.8% in the 32B line case over the conventional DLC, and by 3.4-15.6% in the 64B line case.

3.3 Reducing Static Power

Figure 7 shows the rate of static power of the D-cache to the baseline. The original DLC dramatically decreases power to 0.6-1.3% and 1.2-2.3% for 32B and 64B lines, respectively. Our preactivating DLC cache still keeps quite small power(1.0-2.0% and 1.8-3.9% for 32B and 64B lines, respectively).

In this evaluation, we did not include power consumed in the address prediction. The main component of the address predictor is a table composed of SRAM (10KB SRAM are used for the predictor we used in our evaluation). The SRAM consumes power, but it is negligibly small as predicted in Figure 7 if we compose the table with DLC. The long latency due to DLC is allowed because the pipeline of the current processor is usually deep (at least 16 cycles are consumed from instruction fetch to D-cache access in Intel Pentium 4). Our cacti simulation shows the access to the address predictor table requires two cycles, which is short enough for preactivating.

4. CONCLUSIONS

We have proposed a mechanism that suppresses performance degradation when applying DLC to a cache. Our mechanism predicts the address of the location that a memory instruction will access early in a pipeline, and preactivates a cache line. When the memory instruction actually accesses the cache line, it is already activated, imposing no penalty if the prediction was correct. This latency hiding mechanism minimizes performance degradation due to DLC while keeping the benefit of low power. Our evaluation shows preactivating successfully suppresses performance degradation to an acceptable level with little increase of power.

5. REFERENCES

[1] D. Burger et al. The SimpleScalar Tool Set, Version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin-Madison, June 1997.

[2] M. Horowitz. Timing Models for MOS Circuits. Technical Report SEL83-003, Integrated Circuits Laboratory, Stanford University, 1983.

[3] H. Kawaguchi et al. Dynamic Leakage Cut-off Scheme for Low-Voltage SRAM's. In *Symposium on VLSI Circuits Digest of Technical Papers*, pages 140–141, June 1998.

[4] T. Kuroda et al. A 0.9-V, 150-MHz, 10-mW, 4mm^2, 2-D Discrete Cosine Transform Core Processor with Variable-Threshold-Voltage (VT) Scheme. *IEEE Journal of Solid-State Circuits*, 31(11):1770–1779, November 1996.

[5] MIPS Technologies, Inc. *MIPS R10000 Processor User's Manual, Version 2*, 1996.

[6] S. Mutoh et al. 1-V Power Supply High-Speed Digital Circuit Technology with Multi Threshold-Voltage CMOS. *IEEE Journal of Solid-State Circuits*, 30(8):847–854, August 1995.

[7] G. Reinman et al. Extensions to CACTI. Unpublished document.

[8] G. Reinman et al. Predictive Techniques for Aggressive Load Speculation. In *Proc. MICRO-31*, pages 127–137, December 1998.

[9] K. Wang et al. Highly Accurate Data Value Prediction using Hybrid Predictors. In *Proc. MICRO-30*, pages 281–290, December 1997.

[10] S. J. E. Wilton et al. An Enhanced Access and Cycle Time Model for On Chip Caches. Technical Report 93/5, Digital Equipment Corporation, Western Research Laboratory, July 1994.