Leakage Power Modeling and Reduction with Data Retention *

Weiping Liao
EE Department
UCLA, CA 90095

Joseph M. Basile
Intel Corporation
Santa Clara, CA 95054

Lei He
EE Department
UCLA, CA 90095

Abstract

In this paper, we study leakage power reduction using power gating in the forms of the Virtual power/ground Rails Clamp (*VRC*) and Multi-threshold CMOS (*MTCMOS*) techniques. We apply power gating to two circuit types: memory-based units and datapath components. Using a microarchitecture-level power simulator, as well as power and timing models derived from detailed circuit designs, we further study leakage power modeling and reduction at the system level for modern high-performance VLIW processors. We show that the leakage power can be over 40% of the total power for such processors. Moreover, we propose time-out scheduling of VRC to reduce power up to 85.65% for L2 cache. This power savings results in close to 1/3 total power dissipation for the VLIW processors we study.

1. INTRODUCTION

As semiconductor technology scales down, the leakage power increases exponentially because the transistor threshold voltage is reduced to compensate for the performance loss caused by the scaling down of the supply voltage. It is anticipated that the leakage power will soon become comparable to the dynamic power. In this paper, we study leakage power modeling and reduction at both circuit and (on-chip) system levels. We consider power gating via the Virtual power/ground Rails Clamp (*VRC*) and Multi-threshold CMOS (*MTCMOS*) techniques shown in Figure 1. In Section 2, we evaluate VRC and MTCMOS using circuit-level SPICE simulation and develop power and timing models for two types of circuits: memory array structures and datapath components. In Section 3, we study system-level leakage power reduction using power gating, and we propose time-out-based VRC for L2 cache. We show that our approaches can reduce total power by about 1/3 for modern high-performance VLIW processors, based on a cycle-accurate microarchitecture-level simulator and power/timing parameters developed in Section 2. Conclusions and future work are provided in Section 4.

Related work in the literature includes: [1] evaluated circuit-level power and timing characteristics of the three leakage power reduction techniques: input vector control, body bias control, and power supply gating using phase locked loops (PLLs). As pointed out in [1], PLLs are suitable for global power gating but not for the finer granularity of power gating to be studied in this paper. [2] employed MTCMOS to reduce leakage power for integer units in VLIW processors and exploited scheduling slacks among instruction bundles to increase the chance of power gating via MTCMOS. This study is orthogonal to our algorithms in section 3; therefore, all approaches can be simultaneously applied. Cache power reduction is critical for modern processors. [3] presented the first integrated circuit/architecture approach to reduce instruction cache power using MTCMOS. However, [3] did not consider the non-negligible wake-up time of a sleeping cache and extra compulsory cache misses due to cache content loss introduced by MTCMOS. [4] proposed a power-efficient cache design by adaptively turning off cache lines. VRC was assumed in that approach with data retention. Same as [3], [4] did not consider the impact from the VRC wake-up time, although part of its effect can be compensated by the adaptive algorithm. Further, [4] assumes the power savings is proportional to the turn-off ratio. Our work shows (see Table 4) that simply counting the turn-off ratio or a component's utilization rate is not accurate for power estimation, since VRC has non-negligible extra transition energy. In Section 3.3, we apply time-out-based VRC to reduce cache power with cache content retention and minimize cache misses by considering the realistic wake-up time of VRC. Further, [3] and [4] both applied power reduction techniques to L1 cache. In this paper, we show that a more power-efficient scheme is to throttle the L2 cache, but not the L1 cache.

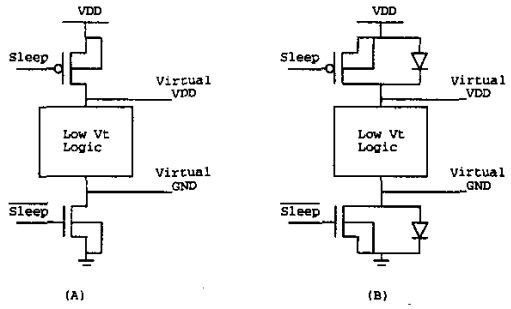

Figure 1: Power gating via (a) MTCMOS and (b) VRC.

*This research was partially supported by SRC contracts 782 and 1008, and a grant from Intel. We used computers donated by HP and SUN Microsystems. Mr. Basile's research was performed while he was a student at University of Wisconsin, and Dr. He was a professor at University of Wisconsin. Comments can be addressed to lhe@ee.ucla.edu.

0-7803-7607-2/02/$17.00 © 2002 IEEE

Circuit Type	Leakage Reduction Technique	Delay % Change	Leakage Power % Reduction	Retention Voltage (V_{diff})	Stdby-Inact. Time (t_{s-i})	Inact.-Stdby Time (t_{i-s})	Min. Idle Time
SRAMs	MTCMOS	3.82% ↑	99.94% ↓	-	0.4ns	3ns	$< 5ns$
SRAMs	VRC	3.25% ↑	99.8% ↓	0.138V	0.5ns	4ns	$> 10ns$
Adders	MTCMOS	6.14% ↑	99.99% ↓	-	0.3ns	1ns	$< 5ns$
Adders	VRC	12.8% ↑	99.8% ↓	0.396V	0.4ns	1.5ns	$> 10ns$

Table 1: Performance, leakage reduction, and data retention results for MTCMOS and VRC.

2. CIRCUIT-LEVEL LEAKAGE POWER REDUCTION

In this section, we study leakage power reduction at the circuit level. We begin by describing MTCMOS and VRC, which both implement power gating to reduce leakage power. We then develop power/timing models for MTCMOS and VRC, followed by a comparison of the two techniques in terms of power, timing, and applicability to datapath and memory circuits.

2.1 MTCMOS and VRC

Circuits that feature the use of power gating exhibit three operating modes: (i) *active mode*, in which a circuit performs an operation and dissipates both dynamic and static power, defined as active power (P_a); (ii) *standby mode*, in which it is active but idle and waiting to execute an operation, dissipating only static leakage power defined as standby power (P_s); (iii) *inactive mode*, in which a circuit is deactivated by power gating, dissipating a reduced static leakage power defined as inactive power (P_i). A circuit in standby mode can be readily used, but it takes a non-negligible amount of time for a circuit in inactive mode, also known as *sleep mode*, to wake up and perform an operation.

MTCMOS [5] uses high-V_t sleep transistors connected to VDD and GND, with the logic in between implemented by low-V_t transistors (see Figure 1(a)). In the active and standby modes, for which the sleep transistors are turned on, the virtual VDD and GND rails function as the actual rails. In inactive (sleep) mode, the large leakage current of the low-V_t logic is greatly reduced by the gating of the power supplies. However, remaining in inactive mode for a few hundred cycles may result in the loss of stored state for memory units and flip-flops, making MTCMOS suboptimal if data retention is desired.

VRC [6] addresses and solves the problem of data retention by placing diodes across the sleep transistors for VDD and GND, as shown in Figure 1(b). In the active and standby modes, virtually all current flows through the sleep transistors, which are turned on. At the switch to inactive mode, these transistors are turned off, a small current flows through the diodes, the virtual VDD level decays from VDD, and the virtual GND level rises from GND. However, the virtual level changes are clamped by the built-in potential of the diodes, maintaining a voltage difference that allows data to be retained in SRAM arrays. The diodes are standard IC-based diodes that require no special processing. However, they result in a larger area overhead for VRC compared to MTCMOS.

In order for a leakage reduction technique to be worth the effort, the energy dissipated with it applied, E_{w_tech}, must be less than that without it applied, E_{w/o_tech}. These energies are defined by [1]:

$$E_{w/o\_tech} = P_s * t_{idle} \quad (1)$$

$$E_{w\_tech} = P_i * (t_{idle} - t_{s-i} - t_{i-s}) + E_{s-i} + E_{i-s} \quad (2)$$

Typically, leakage power reduction techniques have a dynamic power overhead that is dissipated during both the standby-inactive and inactive-standby transitions. The energy associated with this overhead is denoted by E_{s-i} and E_{i-s} in (2). To compensate for this extra power, the circuit must remain in inactive mode long enough to offset it. The *minimum idle time* (*M.I.T.*) required to achieve energy savings with the leakage reduction technique is calculated by [1]:

$$M.I.T. = \frac{E_{s-i} + E_{i-s} - P_i * (t_{s-i} + t_{i-s})}{P_s - P_i} \quad (3)$$

2.2 Technique Comparison and Power Models

For our MTCMOS/VRC comparison and power model development, we chose two types of circuits: conventional SRAM arrays to represent memory, and static CMOS carry-lookahead adders to represent datapath components. SPICE simulations were performed to obtain the active, standby and inactive power for each type of circuit, using the 0.10μm Berkeley Predictive Technology Model (BPTM) transistor models [7] and a VDD of 1.3V, as specified by the International Technology Roadmap for Semiconductors (ITRS).

In general, both PMOS and NMOS sleep transistors are not required for MTCMOS and VRC, as one of either type can achieve the desired effect. In our experiments, we used NMOS sleep transistors for MTCMOS, and both PMOS and NMOS sleep transistors for VRC. Regarding technique granularity, 256-cell SRAM subblocks and 4-bit adder subblocks were used. Experiments showed that a diode area of 1% of the total area of all transistors in the leakage control subblock and a sleep transistor area of 5% of the respective total PMOS or NMOS area provide the best tradeoff between area penalty, circuit delay, data retention, and leakage reduction. Additional experimental settings are described in [8].

Table 1 shows the circuit delay, leakage reduction, data retention, and transition time results obtained by applying MTCMOS and VRC to SRAM arrays and adders. For the SRAM arrays, circuit delay is based on their read and write times. As shown, MTCMOS has a greater adverse effect on circuit delay than VRC, due to slower reduction of the bitline voltage levels during read and write operations. The circuit delay for the adders, which is based on calculation times, favors MTCMOS over VRC because VRC's additional sleep transistor degrades the pullup and pulldown response times of the logic. For both SRAMs and adders, the percentage of leakage power reduction is above 99.9% for MTCMOS and

constant at 99.8% for VRC.

V_{diff} is the voltage difference between the virtual VDD and GND rails with power gating applied in inactive mode, which retains the data stored in SRAM array cells. Despite the low V_{diff} of 0.138V for SRAMs with VRC, as shown in Table 1, it was verified that the values stored in the SRAM cells are restored to their correct value upon return to standby mode. Of course, this does not consider noise, including soft errors. For adders, the value of V_{diff} is irrelevant because they have no need for a data retention voltage. The important transition time in Table 1 is the inactive-standby time, t_{i-s}, which is the time for the virtual VDD and/or GND rails to stabilize at their full values, allowing normal operation to resume. For both SRAMs and adders, t_{i-s} is greater for VRC than for MTCMOS. Also, M.I.T.s are greater for VRC, due to higher inactive-standby energy overhead, E_{i-s}.

Based on our SPICE simulations for SRAMs with different sizes, we propose the following power models:

$$\begin{aligned} P_d[\text{mW}] &= 4.78619232e-4*(\text{words}) + \\ &\quad 2.5116626e-2*(\text{word\_size}) \quad (4) \\ P_s[\text{mW}] &= (7.99611e-2*(\text{words})*(\text{word\_size}) \\ &\quad +0.182395*(\text{words}) \\ &\quad -0.26431*(\text{word\_size})) \\ &\quad *8.9322e-4 \quad (5) \\ P_{i\_V}[\text{mW}] &= (8.051544e-4*(\text{words})*(\text{word\_size}) \\ &\quad +0.2843268*(\text{words}) \\ &\quad -0.0130176*(\text{word\_size})) \\ &\quad *3.2074e-5 \quad (6) \\ P_{i\_M}[\text{mW}] &= (7.862413e-4*(\text{words})*(\text{word\_size}) \\ &\quad +0.0875215*(\text{words}) \\ &\quad -0.00981985*(\text{word\_size})) \\ &\quad *3.2074e-5 \quad (7) \end{aligned}$$

where the *words* and *word_size* variables are the number of rows and columns, respectively, for a SRAM array. Note that Equation (4) is for the dynamic power (P_d). The active power P_a is the sum of dynamic power P_d and standby power P_s. P_d is the average of the active read and write power. Both operations affect only one word and all other words dissipate static leakage power during the operation, so the value of *word_size* affects P_d much more than the value of *words*. Correspondingly, the second term in Equation (4) dominates. The standby power, P_s, and the inactive power with MTCMOS and VRC, which are P_{i_M} and P_{i_V}, are not simply proportional to the total number of cells because of the power contributed by such extra circuits as wordline drivers and precharge transistors.

Based on our simulations for different adder sizes, we propose the following power models:

$$\begin{aligned} P_a[\text{mW}] &= 0.01764122*(\text{adder size}) + 0.14518 \quad (8) \\ P_s[\text{mW}] &= (0.77136*(\text{adder size}) \\ &\quad +0.1541536)*8.9322e-4 \quad (9) \\ P_{i\_V}[\text{mW}] &= (0.06489864*(\text{adder size}) \\ &\quad +6.0111)*3.2074e-5 \quad (10) \\ P_{i\_M}[\text{mW}] &= (0.05673603*(\text{adder size}) \\ &\quad +2.6938)*3.2074e-5 \quad (11) \end{aligned}$$

With the assumption that each calculation takes one clock cycle, the adder P_a increases linearly with the adder size. As shown in Equations (9) through (11), P_s, P_{i_M}, and P_{i_V} are linearly proportional to the adder size. Further, we assume the temperature is 110C in Equations (4) through (11).

Based on the results in Table 1, MTCMOS and VRC have comparable performance and leakage power reduction for SRAMs, but the ability of VRC to retain its data in inactive mode easily offsets its higher M.I.T. On the other hand, since adders have no need for a data retention voltage, the higher M.I.T. for VRC does not justify its use over MTCMOS for adders. Therefore, in our subsequent microarchitecture-level experiments, we chose VRC as the leakage power reduction technique for memory-based components and MTCMOS as the technique for arithmetic datapath components.

3. SYSTEM-LEVEL LEAKAGE POWER REDUCTION

In this section, we study leakage power reduction at the system level for VLIW processors. We first present our power simulator and system configuration, then discuss various power gating schemes including ideal power gating, selective way method, and time-out scheduling of VRC for caches. In our technical report [8], we also study leakage reduction for integer and floating-point units with MTCMOS.

3.1 PowerImpact and System Configuration

We have integrated the power model discussed in the last section into the IMPACT toolset [9] for an EPIC/VLIW architecture, and we named the resulting new toolset PowerImpact [10]. We denote the BTB, L1 instruction cache (IL1), L1 data cache (DL1), L2 cache, and register file (REG) as *memory-based* components, and we denote the decoder (including issue logic), integer components (INT), and floating-point components (FP) as *memory-less* components. We apply VRC to the memory-based components and MTCMOS to the memory-less components. For each component, there are three energy stages: active (P_a), standby (P_s), and inactive (P_i). The power consumption E of a component is calculated by

$$E = P_a \cdot C_a + P_s \cdot C_s + P_i \cdot C_i + E_{overhead} \quad (12)$$

where C_a, C_s, and C_i are active cycles, standby cycles, and inactive cycles, respectively, and are counted by the cycle-accurate simulator in PowerImpact. Further, $E_{overhead}$ is the total transition energy when turning on/off components. It is the product of the number of times the component has been throttled (given by simulator) and the sum of E_{i-s} and E_{s-i} from our circuit-level simulation.

We summarize the system configuration used in our experiment in Table 2, with the system power distribution and the minimum idle times (M.I.T.s) in Table 3. We assume that the clock frequency is 2GHz, and we derive M.I.T.s and inactive-to-standby transition times according to Table 1.

The power consumption values P_a, P_i, and P_s for memory-based components can be calculated according to Eqns. (4)-(7). For the caches and the BTB, we first partitioned the data and tag sections into pieces of SRAM arrays by using the CACTI 3.0 toolset [11]. Then we applied Eqns. (4)-(7) for power consumption of each individual SRAM array. Finally, we added them together for the whole system power consumption. The power consumption values for REG were

Component	Configuration			
Decode (MTCMOS)	6-issue width			
BTB	512 entries 4-way associative, Two-level predictor, Automaton_A3 counter type			
Register file	128 integer and 128 floating-point registers with 64-bit data width			
Memory	page size 4096 bytes, latency 30 cycles			
Memory Bus	8 bytes/cycle			
ALU (MTCMOS)	Number	Latency		
Integer	4	1 cycle for add, 2 cycles for multiply and 15 cycles for division		
Floating-point	2	2 cycles for add/multiply, 15 cycles for division		
Cache (VRC)	Size	Block size	Associativity	Policy
L1 Instruction	32 KB	32 bytes	2	LRU
L1 Data	32 KB	32 bytes	4	LRU
L2	512 KB	64 bytes	8	LRU

Table 2: System configuration for experiments.

Component	P_a (mW)	P_s (mW)	P_i (mW)	M.I.T.	T_{i-s}
Decode	126.12 (10.99%)	4.46	0.02	1	6
BTB	61.96 (5.40%)	25.93	0.61	30	8
Integer ALU	126.12 (10.99%)	4.46	0.02	1	6
FPU	126.12 (10.99%)	4.46	0.02	1	6
I-L1 Cache	79.92 (6.96%)	19.84	0.018	125	8
D-L1 Cache	88.67 (7.73%)	19.80	0.0167	128	8
L2 Cache	534.17 (46.55%)	309.53	0.193	114	8
Registers	4.52 (0.39%)	1.18	0.0054	119	8

Table 3: Power-related parameters. The *M.I.T.* is the minimum idle time in cycles, and t_{i-s} is the inactive-standby transition time in cycles. The percentage value in the column of P_a is the percentage of each component's contribution to total processor power. We assume a 2GHz clock frequency and 0.10μm technology.

based on Eqns. (4)-(7) directly.

Because P_a of the decoder, INT, and FP cannot be calculated from previous equations, we derive their values as follows: We first calculate the power of the L1 cache in the DEC 21264 memory hierarchy as described above, then compute P_a for the caches, decoder, INT, and FP according to the following published power ratio in DEC 21264 [12]:

$$P_a(caches) : P_a(INT) : P_a(FP) : P_a(Decode) = 3 : 2 : 2 : 2.$$

We chose the DEC 21264 because it has a system configuration similar to that used in our experiment. Further, we obtained P_i and P_s of the decoder, INT, and FP according to the ratio between P_a, P_i and P_s given in Eqns. (8)-(11).

3.2 Ideal Power Gating

For ideal power gating, it is assumed that we can schedule a power gating event *in time* for any idle period longer than the M.I.T. to maximize power savings, and we can wake up a component *in time* to avoid performance loss. We divide Decode and REG into six units, INT into four units, and FP into two units, and we turn on/off each unit *independently*. Similarly, we independently turn on/off one cache way in a multi-associativities cache. Since the ideal scheduling assumes the unit can be turned on in time when it is required, there is no change in cache behavior. Further, we assume that we can schedule a clock gating event *in time* for any idle period shorter than M.I.T. (to reduce dynamic power). Therefore, our ideal power gating combines both clock and power gating, and it is also donated as clock/power gating in this paper. Note that clock gating is not applicable to memory units except for their control logic parts. The power savings of ideal power gating can be calculated by analyzing the trace of usages of each component, providing a theoretical upper bound of the leakage power reduction without losing any performance.

	No throttling	Clock gating	Clock/Power Gating	Util
go	100%	52.62%	9.93%	5.78%
ijpeg	100%	55.04%	15.36%	11.07%
equake	100%	53.07%	15.06%	9.95%
art	100%	51.28%	18.98%	7.03%

Table 4: Whole system energy with superblock formulation and ideal scheduling. *go* and *ijpeg* are integer benchmarks, and *equake* and *art* are floating point benchmarks. The column of *Util* shows the energy consumption based only on utilization rate of components.

We compare power dissipation in Figure 2 for (1) no throttling; (2) (ideal) clock gating and (3) (ideal) power gating (with clock gating). For each component, the total energy with no throttling is 100% and the other two cases are normalized to the no-throttling case. It is easy to see that power gating is effective to reduce power for the BTB, L1 data cache, INTs, FPUs, and L2 cache, and it obtains the largest

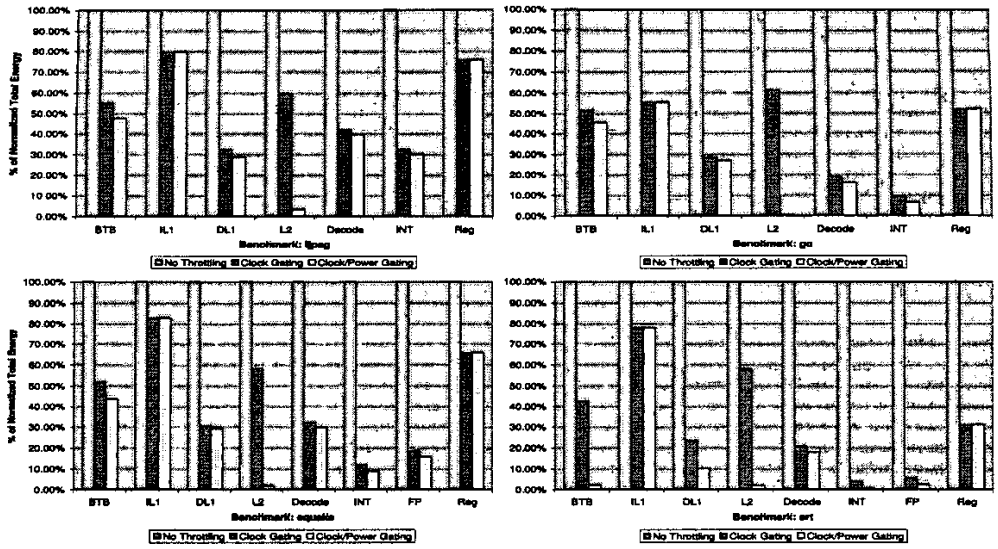

Figure 2: Leakage power reduction under superblock formation and ideal scheduling.

power reduction for L2 cache. In Table 4, we combine the power of different components and compare the total power of the entire processor. Compared to no throttling, the total power can be reduced to 51.28% and 9.93% by using ideal clock gating and ideal power gating, respectively. In other words, ideal power gating achieves up to 90.07% total power reduction. The gap between power savings values with clock gating and power gating is a lower bound of the total leakage power for a system. Table 4 indicates that the total leakage power is over 40% for the modern high-performance VLIW processor specified in Table 2.

Table 4 also shows our experimental results based on the utilization rate of components; i.e., a component only consumes P_a when it is used and P_i when it is idle and powered down. The energy reduction is more than with ideal power gating; it is a theoretical upper bound of power savings. Therefore, simply calculating leakage energy based on utilization rate may exaggerate the leakage reduction and lead to an errant conclusion.

Due to relatively large inactive-standby transition times, t_{i-s}, it is obviously difficult to achieve ideal power gating. Below, we propose time-out scheduling of VRC for L2 cache, as L2 cache consumes the most power among all components.

3.3 VRC Scheduling for Caches

Caches consume a large portion of processor energy and have become the focus of energy reduction research in literature. Existing research on caches mainly focuses on L1 cache. However, having large on-chip L2 caches is a trend in recent high-performance processor designs. As a result, the energy consumption of L2 cache becomes dominant. Because throttling of L1 cache will inevitably increase the L1 cache miss rate and increase accesses and energy consumption of L2 cache, it is not clear whether throttling L1 cache is beneficial from the system point of view. Table 5 shows the energy reduction result by applying the selective way algorithm [13] to L1 cache in our cache hierarchy. This algorithm dynamically turns on/off one or more cache ways based on the access frequency in a given time window. One can easily see that even though a large reduction of L1 cache energy is achieved, the reduction of total L1 and L2 cache energy is negligible. Because L2 cache consumes much more power than L1 cache, and L2 cache exposes more chances for throttling (see Figure 2), we focus on L2 cache in this paper.

benchmark	L1 energy	L1 + L2 energy
equake	82.62%	99.00%
go	88.35%	99.70%

Table 5: Energy consumption when the selective way algorithm is applied to L1 data cache. The value is normalized to the energy with only clock gating. The performance loss is bounded within 1%.

Because the miss penalty of L2 cache is high, a slight increase of miss rate will lead to a much worse performance degradation for L2 cache than for L1 cache. Therefore, the selective way algorithm and cache re-sizing algorithm [3] originally developed for L1 cache are not suitable to reduce L2 cache energy, as shown in our initial experiments. In this paper, we propose the following Fix Time-Out scheme (*FTO*) scheme and Self-Adaptive Time-Out (*SATO*) scheme, where the *entire* L2 cache is throttled as a single unit.

In time-out schemes, the L2 cache is turned off by VRC if there is no L2 cache access for longer than a given time threshold. If a L2 cache request comes when the L2 cache is inactive, a cache miss (called *VRC* miss) occurs and the cache will be waken up immediately. The wake-up time (i.e., t_{i-s}) is 8 cycles, according to Table 3. In the *FTO*

scheme, the time-out threshold is fixed. Further, our experiment showed that using the M.I.T. as the time-out threshold achieves the best performance and power product.

```
T = T0, the minimum idle time of L2.

For each VRC miss
    T = cycles between the last two L2 accesses;

For each time window
    if (VRC_miss < total_accesses * error_bound)
        T = T - 1;
    else if (T < T0)
        T = T + 1;
```

Figure 3: Self-adaptive Time-out Algorithm.

In the *SATO* scheme, the time-out threshold is dynamically decided by the following algorithm (see Figure 3): The initial time-out threshold is the minimum idle time T_0 of L2 cache. When there is a VRC miss, the time-out threshold is set to the cycles between the last two consecutive cache accesses. Further, if the number of VRC misses during a time window W exceeds a given portion (*error_bound*) of total accesses, the time-out threshold is increased by one cycle (but is never larger than T_0); otherwise, it is decreased by one cycle.

Our study in [8] shows that *error_bound* = 1/64 and window size W = 512 cycles achieve the best energy reduction in our experiments. Such a window size is mainly decided by the inactive-standby (wake-up) time of VRC, and it is much smaller than the window sizes (e.g., 100K cycles) used in [13, 3]. A small window enables us to accommodate "real time" changes of cache behaviors in order to keep the cache miss rate as low as possible.

Energy	no throttling	FTO	SATO
equake	100%	55.76%	55.44%
ijpeg	100%	72.07%	68.99%
go	100%	15.47%	14.35%

Table 6: Energy consumption comparison between no throttling (equivalent to clock gating for cache), *FTO* scheme and *SATO* scheme. Performance losses are bound to 1%.

In Table 6, we present the energy consumption results for time-out schemes. Obviously, the *SATO* scheme gets better energy savings than the *FTO* scheme. Compared to no throttling, the total power of L2 cache is reduced to from 72.07% to 15.47% for *FTO*, and from 68.99% to 14.35% for *SATO*. In other words, the power reduction is up to 85.65%. With respect to the L2 power contribution to total processor power in Table 3, this reduction is about 1/3 of the total power dissipation.

4. CONCLUSIONS

In this paper, we have studied leakage power modeling and reduction considering power gating in the forms of *VRC* and *MTCMOS*, with VRC featuring data retention. We have shown that the leakage power can be over 40% of the total power for modern high-performance VLIW processors, and have proposed effective approaches to reduce L2 cache power by up to 85.65%, which is about 1/3 of the total power dissipation.

5. ACKNOWLEDGMENTS

The authors would like to thank Mr. Rakesh Patel and Ms. Wenjie Jiang at Intel for discussions related to VRC during the summer of 2001.

6. REFERENCES

[1] D. Duarte, Y. Tsai, N. Vijaykrishnan, and M. J. Irwin, "Evaluating run-time techniques for leakage power reduction techniques," in *ASP-DAC*, 2001.

[2] W. Zhang, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, D. Duarte, and Y.-T. Fai, "Exploiting vliw schedule slacks for dynamic and leakage energy reduction," in *MICRO 34*, Dec 2001.

[3] S.-H. Yang, M. D. Power, B. Falsafi, K. Roy, and T. Vijaykumar, "An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance I-caches," in *ACM/IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, January 2001.

[4] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptve mode control: A static power efficient cache design," in *10th International Conference on Parallel Architectures and Compilation Techniques*, 2001.

[5] S. Mutoh and *et al*, "1-V power supply high-speed digital circuit technology with multithreshold-voltage cmos," *IEEE Journal of Solid-state circuits*, vol. 30, pp. 847–854, Aug. 1995.

[6] K. Kumagai and *et. al*, "A novel powering-down scheme for low vt cmos circuits," in *1998 Symposium on VLSI Circuits Digest of Technical Papers*, 1998.

[7] U. B. D. Group, *Berkeley Predictive Technology Model (BPTM) 0.10μm SPICE Model Cards*, July 2000.

[8] W. Liao, J. M. Basile, and L. He, "Leakage power modeling and reduction with data retention," *University of Wisconsin Technical Report ECE-02-4, http://eda.ece.wisc.edu/PowerImpact/references.html.*

[9] *http://www.crhc.uiuc.edu/Impact.*

[10] *http://eda.ece.wisc.edu/PowerImpact/.*

[11] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," in *WRL Research Report 2001/2*, 2001.

[12] M. Gowan, L.Biro, and D.Jackson, "Power considerations in the design of the alpha 21264 microprocessor," in *35th Design Automation Conference*, 1998.

[13] D.Albonesi, "Selective cache ways: On-demand cache resource allocation.," in *Proceeding of the MICRO32*, November 1999.