Adaptive Techniques for Leakage Power Management in L2 Cache Peripheral Circuits

Houman Homayoun[‡], Alex Veidenbaum[‡] Jean-Luc Gaudiot[†]
[†]*Department of Electrical and Computer Engineering, UC Irvine*
[‡]*Department of Computer Science, UC Irvine*
{*hhomayou,alexv*}*@ics.uci*.edu {*gaudiot*}*@uci.edu*

***Abstract*— Recent studies indicate that a considerable amount of an L2 cache leakage power is dissipated in its *peripheral* circuits, e.g., decoders, word-lines and I/O drivers. In addition, L2 cache is becoming larger, thus increasing the leakage power.**

This paper proposes two adaptive architectural techniques (*ADM* and *ASM*) to reduce leakage in the L2 cache *peripheral* circuits. The adaptive techniques use the product of cache hierarchy miss rates to guide the leakage control in accordance with program behavior. The result for SPEC2K benchmarks show that the first technique (*ASM*) achieves a 34% average leakage power reduction with a 1.8% average IPC reduction. The second technique (*ADM*) achieves a 52% average savings with a 1.9% average IPC reduction. This corresponds to a *2 to 3 X* improvement over recently proposed static techniques.

I. INTRODUCTION

Static or leakage energy consumption has been growing in both embedded and high-performance processors as transistor geometries shrink. A modern L2 cache occupies a large fraction of the chip area and dissipates a large fraction of chip power, and especially leakage power. A number of architectural and circuit techniques have been proposed to significantly reduce the leakage of the *memory cell array* making *cache peripheral* circuits the main sources of leakage. Recent results have shown that a considerable amount of leakage occurs in the cache *peripheral* circuits, such as decoders, word-lines and output drivers, etc. [8, 12, 17, 24]. This is due to the use of larger, faster and leakier transistors in *peripheral* circuits in order to satisfy timing requirements, while smaller (minimum sized) and less leaky transistors are used in memory cells. In fact, SRAM memory cells can be optimized for low leakage currents without a significant impact on the cell area or the performance [8, 12, 17, 24].

Thus, approaches that concentrate on cell leakage power alone are insufficient and it is extremely important to address leakage in the peripheral circuits. The focus of this paper is therefore to reduce leakage power dissipation in L2 cache, with a particular emphasis on its *peripheral* circuits. Indeed, peripheral circuits use comparatively *large transistors* to drive the associated high loads and meet the memory timing constraints, and thus leakage reduction techniques in these circuits introduce significant additional delays. These delays would significantly increase the L2 cache access time if they are incurred on every access. These leakage reduction techniques are thus not applied in high-performance processors. They are typically used in mobile applications where RAM is put into low-power stand-by mode to reduce leakage current while retaining data. For instance, the Row Decoding scheme [8], reduced both the sub-threshold and gate leakage in peripheral circuits while in sleep mode. How to apply these techniques to L2 cache in high-performance processors is a challenging problem because transitions to and from the low-power mode introduce additional delays. Thus an access issued to an L2, which is in the stand-by mode, will take significantly longer. The main issue is therefore how to achieve power savings without loss of performance.

In a recent work two static architectural techniques, Idle Mode (IM) and Standby Mode (SM), to control sleep signals in the peripheral circuits of L2 caches have been proposed [17]. These techniques exploit two common L2 cache behaviors: the L2 is not accessed because the processor is idle while waiting for the memory response on an L2 miss or because the application has infrequent L2 accesses. In [17] it is shown that IM technique works well in half of the SPECK2K benchmarks but is ineffective for the other half. The SM technique works well in about half of the SPEC2K benchmarks as well, but not for the same benchmarks as the IM [17].

In this work we propose two adaptive architectural techniques for leakage power management in L2 cache. The proposed techniques are adaptive in the sense of making the control decision dynamically and in accordance with program behavior. One of the key contributions of this paper is a metric to guide the adaptive algorithm. The metric is the miss rate product (MRP) of the L1 and L2 miss rates. Intuitively, MRP reflects L2 cache behavior, whether it is idle because the processor is idle waiting for the memory to respond (due to an L2 miss) or because the application has infrequent L2 accesses, driven by the L1 misses. The two adaptive techniques are an Adaptive Static Mode (*ASM*) and an Adaptive Dynamic Mode (*ADM*).

In the *ASM* a decision is made after measuring MRP once after an initial learning period (e.g. after the first 100M committed instructions). In *ADM* the decision is made for a time interval based on the MRP measured in the previous time interval (e.g. every 10M cycles). For both *ASM* and *ADM* techniques, the SM technique is used as the default technique.

In all, this paper makes three major contributions:

1. It demonstrates experimental results, showing that peripheral circuits become the major leakage power

This work was supported in part by the NSF Award CCF-0811882

dissipater in large on-chip cache,

2. It introduces a metric (L1 and L2 miss rate product) which reflects L2 cache behavior, whether it is idle because the processor is idle or because the application has infrequent L2 accesses and,
3. It proposes two adaptive architectural techniques for control of this circuit technique and reduces leakage power.

Our results show a significant reduction in the L2 cache leakage power as well as a significant improvement over the previously proposed static IM and SM techniques. *ADM* reduces L2 leakage power by 52% on average with a 1.9% average performance loss across SPEC2K benchmarks. *ASM* achieves a 34% L2 leakage power reduction with a 1.8% performance degradation, on average.

This paper motivates, describes and evaluates techniques described above targeting the leakage power in the peripheral circuitry of L2 cache. It is organized as follows. Section 2 describes L2 power dissipation details. Related work is described in Section 3. Section 4 presents the leakage control circuit scheme. Section 5 proposes static architectural approach to control the circuit scheme. Section 6 proposes a metric to guide an adaptive leakage control technique. Section 7 describes the two architectural adaptive techniques to control the peripheral leakage. The methodology and experimental results are presented in Section 8.

II. CACHE POWER DISSIPATION

The dynamic power is dissipated in on-chip cache when they are accessed while leakage power is dissipated every cycle. This can make the leakage power exceed the dynamic power in on-chip L2 cache which is accessed infrequently. Figure 1 shows the leakage power breakdown for a 2MB L2 cache components for a 64-bit, 3.4GHz processor (described in Table 1).

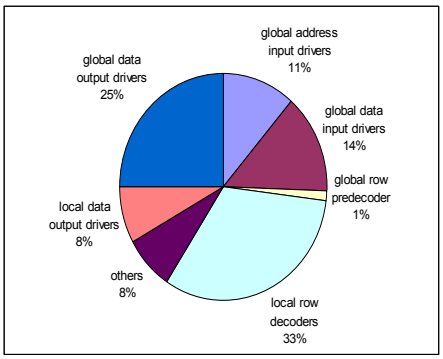

Figure 1. Leakage power of L2 cache components

These results were obtained using CACTI-5 [22] for a 65nm technology. CACTI-5 assumes that the SRAM cell leakage reduction techniques have been applied. Overall, one can see that the peripheral circuits are dissipating more than 90% of total leakage power. The reason is the use of larger, faster and more leaky transistors in peripheral circuits to satisfy the timing requirement, while high V_{th} and less leaky transistors are used in memory cells.

Figure 2 shows the L2 leakage power dissipation as a fraction of the total cache power dissipation (leakage + dynamic) for individual SPEC2K benchmarks. The SPEC2K benchmarks were compiled with the -O4 flag using the Compaq compiler targeted for the Alpha 21264 processor and executed with reference data sets. The architecture was simulated using an extensively modified version of SimpleScalar 4.0 [11]. The benchmarks were fast–forwarded for 3 billion instructions, then fully simulated for 4 billion instructions. The figure shows that the L2 cache leakage power dominates its dynamic power, with an average above 87% of the total.

Table 1. Processor organization

L1 I-cache	128KB, 64 byte/line, 2 cycles
L1 D-cache	128KB, 64 byte/line, 2 cycles, 2 R/W ports
L2 cache	2MB, 8 way, 64 byte/line, 20 cycles
Issue	4 way out of order
Branch predictor	64KB entry g-share,4K-entry BTB
Reorder buffer	96 entry
Instruction queue	64 entry (32 INT and 32 FP)
Register file	128 integer and 128 floating point
Load/store queue	32 entry load and 32 entry store
Arithmetic unit	4 integer, 4 floating point units
Complex unit	2 INT, 2 FP multiply/divide units
Pipeline	15 cycles (some stages are multi-cycles)

Thus it is very important to address the leakage power in peripheral circuits which are responsible for a large fraction of the cache overall power.

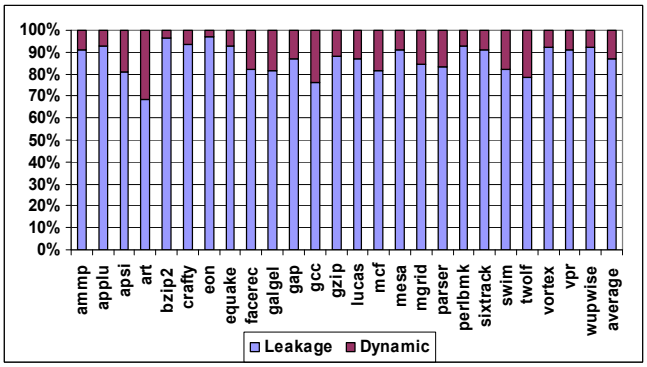

Figure 2 Leakage power as a fraction of the total L2 power dissipation.

III. RELATED WORK

CMOS scaling leads to significant leakage power in the sub-nanometer MOSFET technology [3, 5, 7]. Many approaches to reduce leakage power have been investigated, at the technology, circuit, architecture and compiler/OS levels.

A. Circuit-Level Leakage Control

Four main circuit schemes have been proposed to reduce the leakage power:

1. *Gated-Vdd* turns off the power by using a high threshold transistor [25]. The advantage of this technique is in reducing the leakage power virtually completely. However, it does not retain the state of the memory cell. Similarly, Vss can also be Gated (Gated V_{ss}).

2. *ABB-MTCMOS* increases the threshold voltage of a

transistor dynamically [6]. The overhead of applying this technique in terms of performance and power makes it inefficient. A variation of ABB-MTCMOS technique is proposed in [4, 14] in which the leakage power is suppressed in the unselected part of cache by utilizing super Vt devices (forward body biasing scheme).

3. *Voltage scaling* reduces the source voltage. As explained in [16], due to short-channel effects in deep submicron processes, voltage scaling reduces the leakage current significantly. Voltage scaling can be done dynamically and combined with Frequency Scaling (DVFS) [10, 20].

4. *Cell bias voltage reduction* in the standby state to reduce cell leakage [12] is a technique shown to reduce cell leakage more compared to source voltage scaling [8].

B. Architectural Techniques

A number of architecturally driven cache leakage reduction techniques have been proposed:

1. Flautner et al. proposed a drowsy cache which reduces the supply voltage of the L1 cache line (instead of gating it off completely) [21]. The advantage of this technique is that it preserves the cache line information but introduces a delay in accessing drowsy lines. However, the leakage power saving is slightly lower than the Gated-V_{dd} and V_{ss} techniques.

2. Kaxiras et al. proposed a cache decay technique which reduces cache leakage by turning off cache lines not likely to be reused [19].

3. Powell et al. proposed applying Gated-V_{dd} approach to gate the power supply for cache lines that are not likely to be accessed [13]. This technique results in the data loss in the gated cache line. This leads to an increase in the cache miss rate and loss of performance.

4. Nicolaescu et al. [1] proposed a combination of way caching technique and fast speculative address generation to apply the drowsy cache line technique to reduce both the L1 cache dynamic and leakage power.

5. Bai et al. optimized several components of on-chip caches to reduce gate leakage power [2].

The research mentioned above primarily targeted the leakage in the SRAM cells of a cache. Given the results in Figure 1, cache peripheral circuits are even more important to address in L2 cache. A recent work proposed two architectural techniques, IM and SM, to reduce leakage in L2 cache peripheral [17]. IM asserts sleep signal and put L2 cache peripheral into stand-by mode when processor is idle. SM asserts the sleep signal as default and de-asserts it on an L2 access.

IV. The Leakage Control Approach

The technique proposed in this paper aims at putting the cache peripheral circuits into a stand-by, low power mode. This is accomplished by inserting appropriately-sized "sleep transistors" for both V_{dd} and V_{ss} in the peripheral circuits. The SRAM cell leakage is assumed to be controlled by other techniques such as Gated-V_{dd}, in both the baseline and our enhanced architectures (CACTI-5 accounts for this).

This design technique consists in a set of modifications to the standard L2 cache architecture as shown in Figure 3. They include a global input sleep signal SLP to place its peripheral circuits into the low-power (stand-by) mode. The other modification has to do with cache access in the low-power mode: the processor is not always disabled during this time and can generate L2 accesses. Such L1 misses are stored in a delayed-access buffer. The buffer makes it appear as if the processor has issued a request to the disabled cache. The accesses stored in this buffer are sent to the L2 cache after it returns from the low-power mode. It should be noted that the delay-access buffer is small (only 10 entries, for 10 instructions) and that it dissipates very little power.

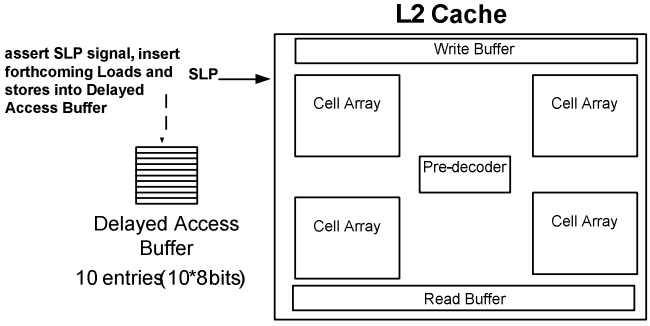

Figure 3. Modified L2 cache architecture

V. Static Architectural Techniques to Reduce Leakage in L2 Cache Peripherals

In this section we briefly discuss how previously proposed static techniques [17], IM and SM, are used to control the sleep transistors in the peripheral circuits. Let us first briefly explain each of these techniques.

A. SM Technique

The static SM technique was proposed by homayoun and veidenbaum [17]. It asserts the sleep signal and puts the L2 cache peripherals into the low-power stand-by mode by default. It only "wakes them up" on an access to the cache. It then keeps the cache in the normal state for J cycles (turn-on period) before returning it to the stand-by mode (SM_J). Any access to the cache made during these J cycles pays no wakeup penalty. A larger J thus leads to lower performance degradation but also lower energy savings.

B. Improved IM Technique

In the IM technique [17] the issue logic and functional units of the processor are checked after a memory access (in our architecture this happens on an L2 miss). It asserts the sleep signal and puts the L2 cache peripherals into sleep mode if the issue logic has not issued any instructions and functional units have not executed any instructions for K consecutive cycles (K=10). Unlike the original IM technique proposed in [17], in this work we de-asserted the sleep signal M cycles before the miss is serviced. This avoids degrading

performance by overlapping the transition from sleep mode to active mode (unlike the performance degradation reported in [17]).

It should be noted that the assumption here is that the memory access latency is deterministic.

Unlike the improved IM, the SM technique degrades performance. Figure 4 shows the IPC decrease with the SM technique for different turn-on periods. While large for SM with small values of J, it becomes comparable or better at J=750 for L2. For one benchmark (*art*) the IPC loss is above 15% for J=500, J=200 and J=100 (not shown).

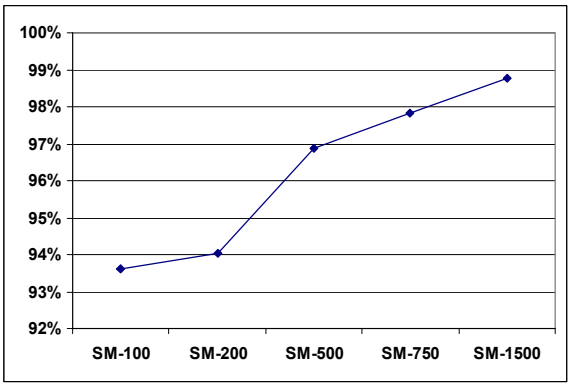

Figure 4. Normalized IPC when SM is applied to L2 for different turn-on periods

Next, we measured the fraction of program execution time during which a cache is in low power mode (Fraction in Low Power mode or FLP) using one of IM or SM.

Figure 5 shows the FLP for the L2 cache using the improved IM technique. In almost half of the benchmarks there is a considerable FLP period. In the rest of the benchmarks, *art, crafty, eon, sixtrack* and *twolf* , the FLP is negligible and there is no leakage reduction opportunity using IM. The reason is that in these benchmarks, the majority of load instructions are satisfied within the cache hierarchy and that the memory accesses are extremely infrequent. The average FLP period is 26.9%.

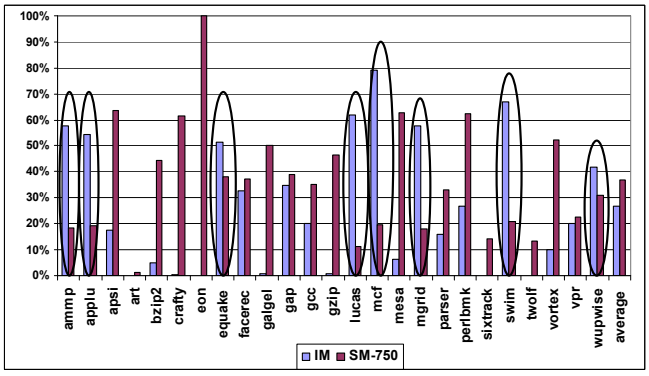

Figure 5. Fraction of time during which the L2 is in low power mode using IM or SM.

Figure 5 also shows the FLP period for the L2 when the SM_750 technique is applied. The average FLP period for SM_750 is 36.6%. The maximum and minimum FLP is for *eon* and *art* respectively. For *eon*, a ray tracing tool, almost all of the load instructions are satisfied in the first level of cache, which explains why the L2 is almost idle. In *art*, an image recognition tool, the L2 is accessed very frequently.

Consequently there is not much opportunity for applying the SM technique.

VI. An Adaptive Approach

From Figure 5, one can make two observations:

1. There are some benchmarks for which SM and IM techniques are both effective; examples are *facerec, gap, perlbmk* and *vpr.*
2. The IM technique works well in almost half of the benchmarks but is ineffective in the other half (highlighted with circles in the figure). The SM techniques work well in about one half of the benchmarks but not the same benchmarks as the IM.

Together, these two observations indicate that an adaptive technique combining the application of the IM and SM techniques and based on the program behavior has the potential to deliver an even greater power reduction in the L2 peripheral circuits. The problem is to decide *which* technique is best and at *what time*. To this end, let us consider the miss rates for the L1 and L2 caches shown in Table 2 for SPEC2K benchmarks. For the L2 to be idle, either there are few L1 misses or many L2 misses waiting for memory. Thus a miss rate product (MRP) of the two levels of cache may be a good indicator of the desired behavior. The (scaled) MRP is shown in the last column of the table.

Table 2. Cache miss rate

	DL1 miss rate	L2 miss rate	L1xL2 miss rates x 10K		DL1 miss rate	L2 miss rate	L1xL2 miss rates x 10K
ammp	**0.05**	**0.19**	**96.11**	lucas	**0.10**	**0.67**	**645.73**
applu	**0.06**	**0.66**	**368.03**	mcf	**0.24**	**0.43**	**1023.88**
apsi	0.03	0.28	75.01	mesa	0.00	0.27	8.02
art	0.41	0.00	0.41	mgrid	**0.04**	**0.46**	**165.13**
bzip2	0.02	0.04	7.09	parser	0.02	0.07	13.76
crafty	0.00	0.01	0.17	perlbmk	0.01	0.46	22.88
eon	0.00	1.00	0.00	sixtrack	0.01	0.00	0.14
equake	**0.02**	**0.67**	**124.36**	swim	**0.09**	**0.63**	**561.41**
facerec	0.03	0.31	86.11	twolf	0.05	0.00	0.16
galgel	0.04	0.01	2.11	vortex	0.00	0.23	6.94
gap	0.01	0.55	38.54	vpr	0.02	0.15	33.95
gcc	0.05	0.04	16.88	wupwise	**0.02**	**0.68**	**122.40**
gzip	0.01	0.05	3.28	Average	0.05	0.31	136.50

One can divide the benchmark in two distinct categories, with high MRP and low MRP. In benchmarks with high MRP, the IM technique should be very effective compared to SM. For instance, in *applu, equake, lucas, mcf, etc.* On the other side are benchmarks with very low MRP, such as *bzip, crafty, eon, etc*, for which SM is more effective compared to IM. The above suggests that one can choose the most effective static control technique, the IM or the SM, based on the MRP.

VII. The Adaptive Techniques

Based on the above discussion, using the MRP to quantify program behavior, two adaptive techniques can be proposed:

A. Adaptive Static Mode (ASM)

In this technique the MRP is measured only once during an initial learning period (the first 100M committed instructions). If the MRP > *A*, a *predetermined threshold,* the IM technique is applied for the remainder of program execution. If the MRP ≤ *A, the* SM_J technique is used. The initial technique is SM_J. Based on MRP values in Table 2 *A*=90 was chosen. This technique is adaptive static in the sense that the decision as to which static technique to use is made only once per program execution and does not change for the rest of program.

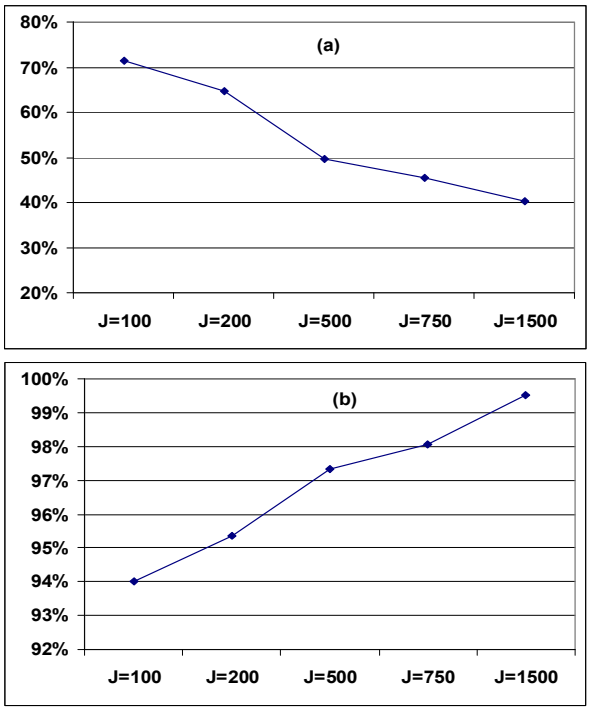

Figure 6. ASM (a) FLP period (b) performance loss

B. Adaptive Dynamic Mode (ADM)

In this technique the MRP is measured continuously during program execution over a K cycle period (K is 10M in this work) and then used to choose the leakage control approach, the IM or the SM, for the next 10M cycles. The following decision making process is used:

If MRP > A the IM approach is used,
if the A ≥ MRP > B the SM_N approach is used, otherwise, the SM_P approach is used.

This technique makes the control decision dynamically and based on program behavior.

In summary, *ASM* attempts to find the more effective static technique per benchmark by profiling a small subset of a program. Then based on the predicted L2 cache behavior it selects one technique and utilizes it for the rest of program execution. *ADM* is *more complex* and attempts to find the more effective static technique at a finer granularity of every 10M cycles intervals based on profiling the previous timing interval.

Figure 6 shows the FLP period and incurred performance loss when *ASM* is applied and for different turn-on periods J. Not unexpectedly, the short turn-on period leads to a larger FLP and consequently more leakage savings. The performance decreases significantly with shorter turn-on period. ASM_750 makes a good power-performance trade-off with a 44% FLP and an approximately 2% performance loss.

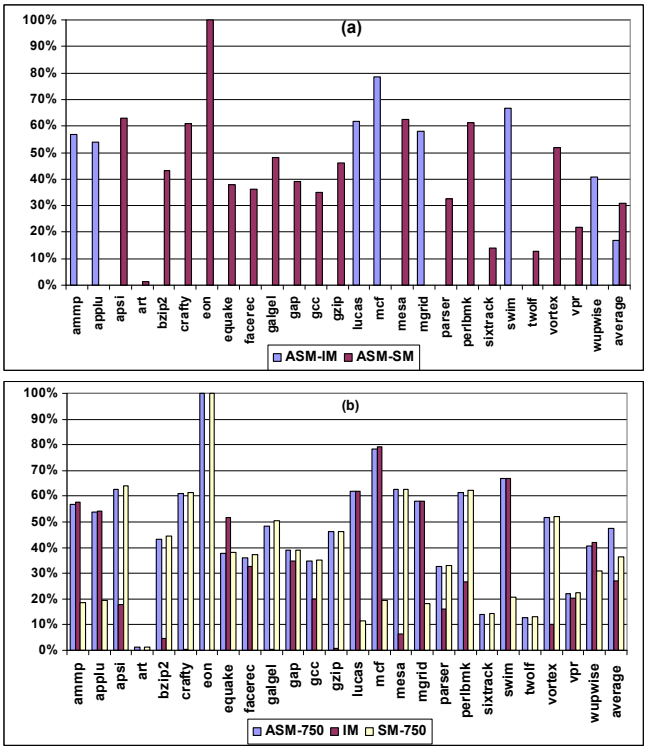

Figure 7. fraction of IM and SM contribution for (a) ASM_750 (b) comparing FLP period of ASM_750 with SM_750 and IM.

To better understand the effectiveness of *ASM* for individual benchmark and in comparison with SM and IM, Figure 7 shows results for ASM_750. Figure 7 (a) shows the fraction of IM (ASM_IM) and SM (ASM_SM) contribution in ASM_750. Figure 7 (b) shows *ASM* compared with non-adaptive static IM and SM_750 techniques. For most benchmarks *ASM* correctly selects the more effective static technique. equake is the exception; the more effective technique is IM while the SM technique is being selected.

These results indicate that for most benchmarks a small subset of program can be used to identify L2 cache behavior, whether it is accessed very infrequently and as such SM is a more effective technique or it is idle since processor is idle and as such IM delivers better results. On average IM contribution in ASM_750 is 16.7% while SM_750 contribution is larger; 30.7%. The average FLP period across most benchmarks is higher than both SM_750 and IM; 47.4% for ASM_750 vs. 36.6% in SM_750 and 26.9 in IM.

Figure 8 shows the normalized IPC for ASM_750 in comparison with SM_750. Not unexpectedly, in ammp, applu, lucas, mcf, mgird, swim and wupwise there is no performance degradation. This is due to the fact that for

these benchmarks the IM technique is selected. For the rest the performance degradation is the same as SM_750. The average performance loss is smaller than SM_750, 1.8% vs 2.2%.

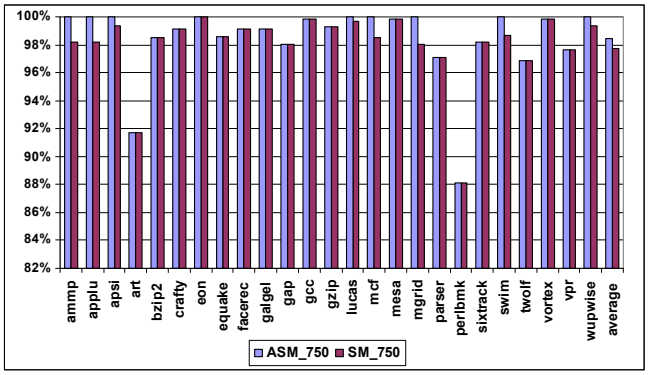

Figure 8. Normalized IPC of ASM_750 and SM_750.

For *ADM* technique there are several parameters that affect the results: the thresholds A and B, turn-on periods N and P in SM_N and SM_P. Due to space limitation we can not present the results for different set of parameters. Our results show that ADM_100_200 for which when MRP is above 90 IM is selected, when it is between 50 and 90 SM_200 is selected and when it is below 50 SM_100 is selected deliver a good power-performance trade-off.

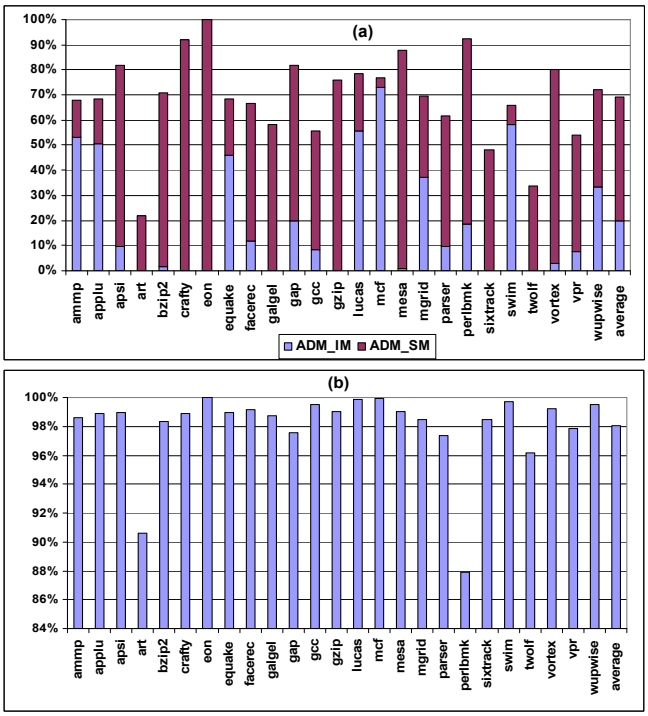

Figure 9. (a) IM and SM contribution and (b) normalized IPC for ADM

Figure 9 (a) shows the fraction of static IM (ASM_IM) and static SM (ASM_SM) contribution in *ADM*. For many benchmarks both IM and SM now make a noticeable contribution. For these benchmarks *ADM* is effective in combining the IM and SM. There are also some benchmarks in which either the IM or the SM contribution is negligible, e.g. art, bzip, crafty and etc. In fact, *ADM* selects the best static technique for these benchmarks. On average, IM and SM contribute 20% and 49.3% respectively.

The total FLP is 69.3%, a ***2x to 3x*** increase over the static IM and static SM techniques. This comes with 1.9% performance degradation on average. The worst performance loss occurs in perlbmk, by 22%.

VIII. POWER AND ENERGY-DELAY RESULTS

This section presents the results for power reduction and energy-delay product for all techniques presented in this work. First, let us describe power assumptions used.

As shown in Section 2, the peripheral circuits controlled by the selected circuit techniques account for 90% of all the leakage power, according to CACTI-5. The power reduction using some of the same leakage control circuits reported in [8, 12] was 88%. These are the values used to obtain power savings shown in Figure 10.

Total dynamic power was computed as N*Eaccess/Texec, where N is the total number of accesses (obtained from simulation), Eaccess is the single access energy from CACTI-5 and Texec is the program execution time. Leakage power computations are similar, but leakage energy is dissipated on every cycle.

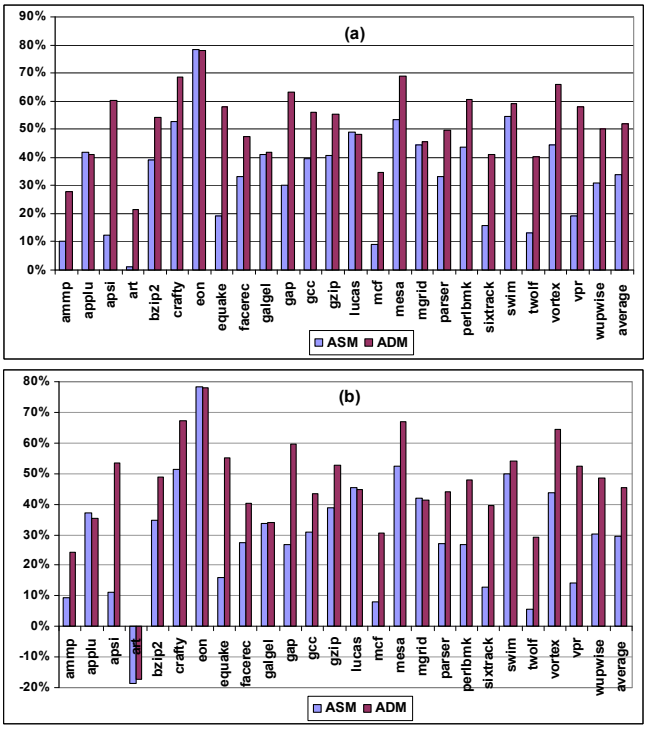

Figure 10. (a) Leakage power savings (b) total energy-delay reduction for all caches

The time delay for transition to/from stand-by mode, STL, used in this work is 10 processor cycles for the L2 cache. This delay is comparable with cumulative cache peripheral delay; word line driver delay, decoder delay, input and output drivers delay. These delays are measured using CACTI and then translated to a 300ps cycle time (the

operating frequency is assumed to be 3.4GHz). The simulator accounts for this delay as required by each technique.

Figure 10 shows a) the leakage power savings and b) total energy delay reduction achieved with techniques discussed in this work.

These results reported are for L2 cache when its peripheral circuits are controlled using ASM_750 or ADM_100_200. On average, leakage reduction for L2 using *ASM* and *ADM* is 34% and 52% respectively.

In most benchmarks a noticeable energy-delay reduction is observed. One exception is art for which the energy-delay product increases by 18 to 19% for the L2 cache. This in fact is due to its significant performance degradation when L2 is controlled with *ADM* or *ASM*. The overall energy delay reduction is 29.4 and 45.5%, respectively, for the L2 using the *ASM* and *ADM*.

IX. Conclusion

This paper presented two adaptive architectural techniques for leakage power management in the L2 cache peripheral circuits. They control the sleep transistors in the peripheral circuitry. The proposed techniques are adaptive in the sense of making the control decision in accordance with program behavior. They exploit L1 and L2 cache miss rate product which intuitively reflect L2 cache behavior, for making the decision dynamically.

The first technique (*ASM*) achieves 34% average leakage power reduction with a 1.8% average IPC degradation. The second technique (*ADM*) achieves 52% average savings with a 1.9% average IPC degradation. This corresponds to a ***2 to 3 X*** improvement over recently proposed static techniques.

References

[1] D. Nicolaescu et al,. Fast Speculative Address Genera-tion and Way Caching for Reducing L1 Data Cache Energy. Proc. *IEEE ICCD*, 2006.

[2] R. Bai et al,. Total leakage optimization strategies for multi-level caches in Proc. *ACM Great Lakes symposium on VLSI*, 2005.

[3] T. Skotnicki et al,.The end of CMOS scaling: toward the introduction of new materials and structural changes to improve MOSFET performance. *IEEE Circuits and Devices Magazine*, Jan.-Feb. 2005, Vol. 21, Issue: 1.

[4] C. H. Kim et al,. A forward body-biased low-leakage SRAM cache: device, circuit and architecture considerations. IEEE Trans. on VLSI Systems, vol. 13, 2005, pp. 349-357.

[5] S. Borkar et al,. Platform 2015: Intel® Processor and platform evolution for the next decade. *Intel Technology Magazine*, March 2005.

[6] K. Nii, et al. A low power SRAM using auto-backgate-controlled MT-CMOS. In *ISLPED*, 1998, pp. 293-298.

[7] M. Bohr. Nanotechnology goals and challenges for electronic applications. IEEE Transactions on Nano- technology, Vol 1, No. 1, March 2002.

[8] Y. Takeyama et al,. A Low Leakage SRAM Macro with Replica Cell Biasing Scheme. *IEEE Journal Of Solid- State Circuits*, Vol. 41, No. 4, April 2006.

[9] F. Hamzaoglu et al,. Analysis of Dual-VT SRAM cells with Full-Swing Single-Ended Bit Line Sensing for On-Chip Cache. IEEE Trans. on VLSI Systems, vol. 10, April 2002,

[10] D. Marculescu. On the use of microarchitecture-driven dynamic voltage scaling. In *Workshop on Complexity-Effective Design*, June 2000.

[11] SimpleScalar4 tutorial, SimpleScalar LLC. http://www.simplescalar.com/tutorial.html.

[12] K. Nii et al., A 90-nm low-power 32 KByte embedded SRAM with gate leakage suppression circuit for mobile applications, IEEE J. Solid-State Circuits, vol. 39, pp. 684-693, Apr. 2004.

[13] M.D. Powell et al,. Gated V_{dd}: A circuit technique to reduce leakage in deep-submicron cache memories. in Proc. *IEEE ISLPED*, 2000.

[14] A. Agarawal et al., DRG-Cache: A Data Retention Gated-Ground Cache for Low Power, DAC 2002. pp. 473-478.

[15] Y. Li et al,. State-preserving vs. non-state-preserving leakage control in caches. in Proc. *IEEE DATE*, 2004.

[16] K. Flautner et al,. Automatic performance setting for dynamic voltage scaling. in Journal of Wireless Networks, pages 260–271, 2001.

[17] H. Homayoun and A. V. Veidenbaum,. Reducing Leakage Power in Peripheral Circuit of L2 Caches, *IEEE-ICCD* 2007.

[18] Bharadwaj S. Amrutur et al,. Speed and power scaling of SRAMs, IEEE Journal of Solid State Circuits. Feb 2000, vol. 35.

[19] S. Kaxiras et al,. Cache decay: exploiting generational behavior to reduce cache leakage power. *IEEE-ISCA, 2001*.

[20] Pentium M processor on 90 nm process with 2-MB L2 cache. Intel Corp. .Jan. 2005. www.intel.com/design/mobile/datashts/302189.htm.

[21] K. Flautner et al,. Drowsy caches: simple techniques for reducing leakage power. *IEEE ISCA*, 2002.

[22] Cacti5, http://quid.hpl.hp.com:9082/cacti/.

[23] S. Rusu et al,. A 65-nm Dual-Core Multithreaded Xeon® Processor With 16-MB L3 Cache, IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 42, JAN 2007.

[24] S. Rusu, H. Muljono and B. Cherkauer,. Itanium 2 processor 6M: higher frequency and larger L3 cache. IEEE Micro, Mar-Apr 2004, Volume: 24, Issue: 2.

[25] M. Powell et al,. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. *IEEE-ISLPED* 2000.

[26] B.S. Amrutur, et al., A replica technique for wordline and sense control in low-power SRAM's, IEEE Journal of Solid-State Circuits, vol. 33, No. 8, Aug.2000.

[27] J. M. Rabaey et al., Digital integrated circuits: a design perspective, Prentice Hall, Second. Edition, 2003.