Multiple Sleep Mode Leakage Control for Cache Peripheral Circuits in Embedded Processors

Houman Homayoun†, Mohammad Makhzan‡, Alex Veidenbaum†

†Center for Embedded Computer Systems
University of California, Irvine, CA
{hhomayou, alexv}@ics.uci.edu

‡Department of Electrical and Computer Engineering
University of California, Irvine, CA
{mmakhzan}@uci.edu

ABSTRACT
This paper proposes a combination of circuit and architectural techniques to maximize leakage power reduction in embedded processor on-chip caches. It targets cache peripheral circuits, which according to recent studies account for a considerable amount of cache leakage. At circuit level, we propose a novel design with *multiple sleep modes* for cache peripherals. Each mode represents a trade-off between leakage reduction and wakeup delay. Architectural control is proposed to decide "when and how" to use these different low-leakage modes using cache miss information to guide its action. This control is based on simple state machines that do not impact area or power consumption and can thus be used even in the resource constrained processors. Experimental results indicate that proposed techniques can keep the L1 cache peripherals in one of the low-power modes for more than 85% of total execution time, on average. This translates to an average leakage power reduction of 50% for 65nm technology. The DL1 cache energy-delay product is reduced, on average, by 20%.

Categories and Subject Descriptors
B.3.2 [MEMORY STRUCTURES], Design Styles: Cache memories; C.1.1 [PROCESSOR ARCHITECTURES], Single Data Stream Architectures: Pipeline processors Systems

General Terms
Design

Keywords
Cache, Leakage Power, Peripheral Circuits, Multiple Sleep Mode, Embedded Processor

1. INTRODUCTION
Static or leakage energy consumption has been growing in both embedded and high-performance processors as transistor geometries shrink. Cache and TLB RAM structures account for a large fraction of processor power consumption [27, 29], and especially of leakage power. A number of process and circuit techniques have been proposed to significantly reduce leakage of the memory cell array making SRAM peripheral circuits the main sources of leakage. Recent results have shown that a considerable amount of leakage occurs in the peripheral SRAM circuits, such as decoders, word-line and output drivers, etc [2, 8, 9, 12, 17, 24]. Figure 1 shows leakage components for different size SRAMs in 65nm technology (based on CACTI 5.1 [22]), with peripheral circuits – data drivers, address driver, decoder and wordline drivers – accounting for over 80% of overall SRAM leakage. The reason is the use of larger, faster and more leaky transistors in peripheral circuits to satisfy timing requirements, while smaller 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 performance [8,12,24].

This paper proposes a combination of circuit and architectural techniques to maximize leakage power reduction in embedded processors. It focuses on caches since they have the highest leakage energy in such processors, and in particular on leakage in SRAM peripheral circuits. There is a large variety of embedded processors, from single-issue, in-order processors with one level of cache to multiple-issue, out-of-order processors with two levels of cache. For this work we define the former as low-end embedded processors, while a single-issue, in-order processor with two cache levels is defined as high-end. Our goal in doing so is to explore leakage reduction in different types of cache hierarchy. The same techniques are applicable to out-of-order embedded processors that also offer other opportunities for cache leakage reduction, but this type of processor is beyond the scope of this paper.

At the circuit level, we utilize our recently proposed approach, *zig-zag share* circuit [2] to reduce the sub-threshold leakage in peripheral circuits of L1 caches. *Zig-zag horizontal and vertical share* technique was shown to be vey effective in reducing leakage of SRAM peripherals. The results in [2] show leakage reduction by up to 100X in deeply pipelined SRAM peripheral circuits, with only a minimal area overhead and small additional delay.

As shown in [2], the wakeup latency of *zig-zag share* could be large, especially in large SRAMs.. To deal with this problem, this paper shows that by increasing the bias voltage of the NMOS footer sleep transistor in *zig-zag share* circuit (and decreasing it for the PMOS header transistor) one can trade leakage reduction

CASES'08, October 19–24, 2008, Atlanta, Georgia, USA.

vs wakeup delay. Thus we propose to use several low-leakage modes with different wakeup times to better control the cache leakage. For instance, one can have a low-leakage mode for an L1 cache with a one-cycle wakeup but it would reduce leakage by only 40%. Alternatively, one can define a mode with a 4-cycle wakeup that saves 90% of leakage. These modes differ only in how they bias sleep transistors and thus can be dynamically switched during execution with almost no delay. The question is when and how to use these different low-leakage modes for L1 caches. Note that this approach can also be applied to L2 caches, but this is beyond the scope of this paper.

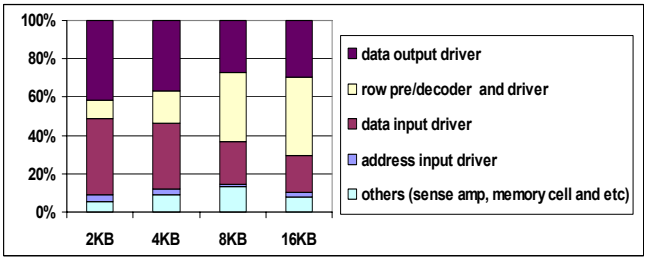

Figure 1. Leakage power component for different cache size

We propose to use architectural control of low-leakage modes in L1 caches. It uses cache miss information to determine its action. The action depends in part on the cache organization and in part on the ability to hide the wakeup delay. In all cases, control is based on simple state machines that do not impact area or power consumption and can thus be used even in low-end processors. Hiding one to four cycles of wakeup latency in a short pipeline typical of embedded processors is difficult in a uniform way, therefore we propose different methods for each delay and low-leakage mode. For instance, one cycle of delay in DL1 access can be completely hidden by starting cache wakeup as soon as instruction type is known in decode.

The most efficient low-energy mode with four cycles of wakeup can be used during cache miss service. Thus we can actually keep the L1 cache peripherals in a basic low power mode with a one-cycle wakeup as default. Other low-power modes can be used when an L2 cache is presented.

The energy savings and performance of various cache configurations for embedded processors are evaluated in this paper using the proposed circuit and architectural techniques. Due to lack of space only L1 data cache results are presented, but it should be clear that the approach can be applied to I-caches as well.

It is shown that our techniques can keep the L1 cache in one of the low-power modes for more than 85% of total execution time, on average. This translates to an average leakage power reduction of 50%. And the energy-delay product is reduced, on average, by 20%.

2. RELATED WORK

A number of techniques were proposed for reducing leakage power at technology, circuit, architecture and compiler/OS levels.

2.1 Circuit-level leakage control

Several circuit techniques proposed to reduce the leakage power in SRAM memories. These techniques were mainly targeting the SRAM memory cell leakage.

The primary technique is voltage scaling which due to short-channel effects in deep submicron processes reduces the leakage current significantly [21]. Another technique is Gated-Vdd which turns off the supply voltage of memory cells by using a sleep transistor and eliminating the leakage virtually completely [23]. However, it doesn't retain the state of the memory cells. The third technique, ABB-MTCMOS, increases threshold voltage of a SRAM cell dynamically through controlling its body voltage [16]. The overhead of applying this technique in terms of performance and area makes it inefficient. Device scaling leads to threshold voltage fluctuation, which makes the cell bias control difficult to achieve. In response, [8] proposed a Replica Cell Biasing scheme in which the cell bias is not affected by Vdd and Vth of peripheral transistors.

[14, 31] proposed a forward body biasing scheme (FBB) in which the leakage power is suppressed in the unselected memory cells of cache by utilizing super Vt devices.

In addition to these four major techniques applied to SRAM memories, there are also leakage reduction techniques in literature which concentrated on generic logic circuits. Examples are sleepy stack [10] and zig-zag super cut-off CMOS (ZSCCMOS) techniques [3, 4]. ZSCCMOS reduces the wakeup overhead associated with Gated-Vdd technique by inserting the sleep transistors in a zig-zag fashion. Sleepy stack proposed to divide the existing transistors into two half size and then insert sleep transistor to further reduce leakage. This approach was shown to be area-inefficient as it comes with 50 to 120% area overhead.

2.2 Architectural techniques

A number of architecturally driven cache leakage reduction techniques have been proposed. Powell et al proposed applying gated-Vdd approach to gate the power supply for cache lines that are not likely to be accessed [13]. Kaxiras et al. proposed a cache decay technique which reduces cache leakage by turning off cache lines not likely to be reused [19]. 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. 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. Zhang et al. proposed a compiler approach to turn off the cache lines for a region of code that would not be accessed for a long period of time [5]. Meng et al presented a perfecting scheme which combines the drowsy caches and the Gated-Vdd techniques to optimize cache leakage reduction [6]. Ku et al. [28] exploit several power density minimization techniques to reduce temperature and further leakage in highly-associative on-chip caches in embedded processors. Due to positive feedback relation of temperature and leakage, they shown on-chip cache leakage reduces significantly.

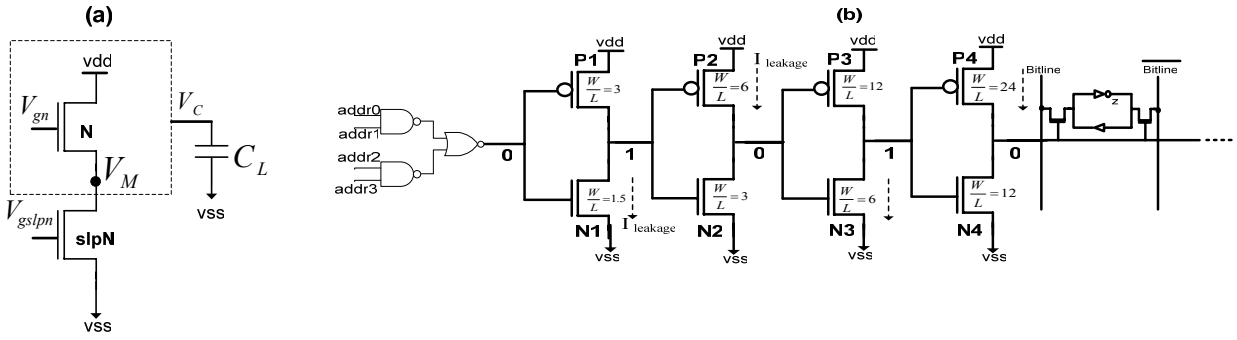

Figure 2. (a) Stacking sleep transistor to reduce leakage (b) Leakage in the wordline driver

In a recent work, Chung et al. proposed a novel approach, which utilize branch prediction assist to selectively wake up only the needed instruction cache line for an embedded processor [30].

All research mentioned above primarily targeted the leakage in the SRAM cells of a cache. Given the results in Figure 1 and recent work such as [2, 8, 9, 12, 17, 24], peripheral circuits are equally if not more important to address in a cache.

3. SLEEP TRANSISTOR STACKING

Stacking sleep transistors have been proposed to reduce sub-threshold (I_{Dsub}) or weak inversion current [32]. IDsub is an inverse exponential function of threshold voltage (V_T). Threshold voltage is a function of Source to Bulk Voltage. An effective way to reduce the leakage of a transistor is by increasing its source voltage (for an NMOS increasing V_{SB}, the source to bulk voltage) [7, 32]. Stacking a sleep transistor (footer NMOS or header PMOS transistor) as shown in Figure 2(a) could deliver this effect. In this figure by stacking transistor N with slpN source to body voltage (V_M) of transistor N increases. When both transistors are off increase in V_M increases the V_T of the transistor N and therefore reduces sub-threshold leakage current. [32]. Size (W/L) and bias (V_{gslpn}) voltage of the stacked sleep transistor determines the V_M [15, 32]. Reducing the sleep transistor bias reduces the leakage while increasing the circuit wakeup period which is the time to pull the VM down to ground. Thus there is a trade-off between the amount of leakage saved and the wakeup overhead [15].

A wordline driver shown in Figure 2(b) increases the gate voltage of the access transistors of all cells connected to the selected wordline. The number and size of inverters in the chain are chosen to meet the timing requirements for charging or discharging the wordline. The size of inverters in the chain decreases from decoder side to the wordline to increase the effective fan-out. The inverter chain has to drive a logic value 0 to the pass transistors when a memory row is not selected. Thus the driver cannot be simply shut down when idle. Transistors N1, N3 and P2, P4 are in the off state and thus they are leaking.

Stacking header and footer sleep transistors with all NMOS and PMOS transistors in the chain reduces their leakage; however, aside from the area overhead, it increases the propagation delay of the inverters in the driver chain followed by an increase in the rise/fall time of the wordline. Rise and fall time of an inverter output is proportional to the R_{peq} * C_L and R_{neq} * C_L, respectively, where R_{peq} is the equivalent resistance of the PMOS transistor, R_{neq} is the equivalent resistance of the NMOS transistor, and C_L is the equivalent wordline output capacitive load [7]. Inserting sleep transistors increases R_{neq}, R_{peq} and thus the rise time and fall time of the wordline driver as well as its propagation delay [2, 7]. While increasing the rise time and propagation delay (due to its impact on access time) is not desirable, increasing the fall time is not tolerable since it can affect memory functionality [18, 20]. Increase in the fall times of the wordline increases the access transistor's active period of a memory cell during a read operation. This results in the bitline over-discharge and the memory content over-charge during the read operation. Such over-discharge not only increases the dynamic power dissipation of bitlines but, more importantly, can cause a memory cell content to flip if the over-discharge period is large [7,20]. In brief, to avoid impacting memory functionality the sense amplifier timing circuit and the wordline pulse generator circuit need to be redesigned. To avoid the redesign of these critical units and, moreover, not to increase bitline dynamic power dissipation we use *zig-zag share* circuit technique proposed in [2].

3.1 zig-zag share circuit

In this approach, sleep transistors are inserted in a zig-zag fashion [3, 4] keeping the R_{peq} of the first and third inverters and R_{neq} of the second and fourth inverters constant. This technique keeps the fall time of the circuit the same as in the baseline circuit with no leakage control. However, the rise time of the circuit is affected by the zig-zag scheme. In addition, using one sleep transistor per inverter logic increases the area for the zig-zag scheme.

To improve both leakage reduction and area-efficiency of the zig-zag scheme, [2] proposed using one set of sleep transistors shared between multiple stages of inverters which have similar logic behavior, such as stage 1 and 3 in a studied chain of inverters. To further reduce leakage power [2] proposed to also share one set of sleep transistors (slpN and slpP) vertically with adjacent rows of a (wordline) driver. Figure 3 shows the *zig-zag horizontal and vertical sharing* circuit (in brief zz-hvs) when two adjacent wordline drivers share one set of sleep transistors. Intuitively, in vertical sharing (for instance for N_{11} and N_{21}), the virtual ground voltage (VM in Figure 3) increases in comparison to when there is no vertical sharing.

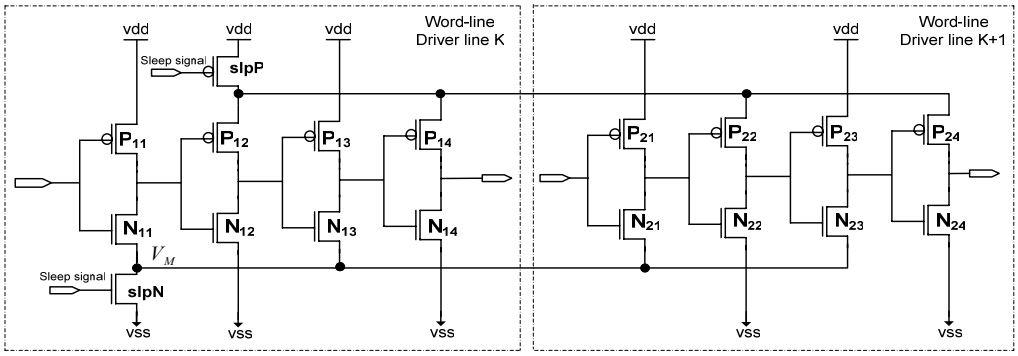

Figure 3. Zig-zag horizontal and vertical sharing circuit

Results in [2] show that using zz-hvs reduces the leakage power significantly, by 10 to 100X, when 1 to 10 wordlines share the same sleep transistors.

4. ZIGZAG-SHARE WITH MULTIPLE SLEEP MODES

As described in Section 3, to benefit the most from the leakage reduction by stacking sleep transistors we need to keep the bias voltage of NMOS footer sleep transistor as low as possible (and for PMOS header transistor as high as possible). The drawback of such biasing is its impact on wakeup latency of the circuit transitioning from sleep mode to active mode which requires the voltage of virtual ground to reach to the ground voltage [2]. Such wakeup delay would significantly impact performance if it was incurred frequently. One way to alleviate the impact of wakeup delay is to control the gate voltage of the sleep transistors (both footer and header) [15]. For instance, increasing the gate voltage of footer sleep transistor (in Figure 2) reduces the virtual ground voltage (V_M) which leads to reduction in the circuit wakeup delay overhead. The negative impact of such biasing is a reduction in leakage power savings. By controlling the gate voltage of footer and header transistors we can thus define different sleep modes where each mode has a different wakeup delay overhead and a different amount of leakage power reduction.

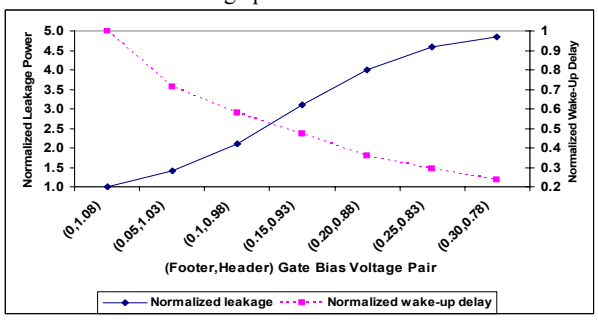

Figure 4. Normalized wakeup delay and leakage power for different pair of footer and header gate bias voltage

The proposed multiple sleep mode *zig-zag share* approach was applied to SRAM wordline driver. A test experiment was set up in which the wordline inverter chain drives 256 one-bit memory cells. The memory cells and wordline drivers were laid out using Mentor Graphic IC-Station in TSMC 65nm technology and simulated using Synopsis Hspice at typical corner (25 º) with extracted netlist and the supply voltage of 1.08V. Empirical results presented are for the normalized leakage power and wakeup delay for different pair of sleep transistors bias voltage.

Figure 4 shows normalized wakeup delay and normalized leakage power for different pairs of footer and header gate bias voltage when zz-hvs is shared by 10 rows of wordline drivers. The figure shows a clear trade-off between the normalized wakeup overhead and leakage power.

Based on these experimental results, four sleep modes were defined. Table 1 shows wakeup delay and relative peripheral circuit leakage reduction for the four different modes. The basic low power mode has the lowest leakage reduction but shortest wakeup delay. Next is lp mode which has higher leakage savings. Aggressive and ultra sleep modes have even higher leakage savings but also a longer wakeup delay.

Table 1. Peripherals multiple sleep mode

power mode	wakeup delay (cycle)	leakage reduction (%)
basic-lp	1	42%
lp	2	75%
aggr-lp	3	81%
ultra-lp	4	90%

Finally, note that the power overhead of waking up peripheral circuits from any low power mode is negligible, almost equivalent to the switching power of sleep transistors. Sharing a set of sleep transistors horizontally and vertically for multiple stages of a (wordline) driver makes the power overhead even smaller.

5. APPLYING ZZ-HVS TO L1 DATA CACHE

This section describes the architectural approach to control the zz-hvs sleep transistors in DL1 cache for two different types of embedded processors: a low-end and a high-end. We start by briefly describing the processor and the experimental

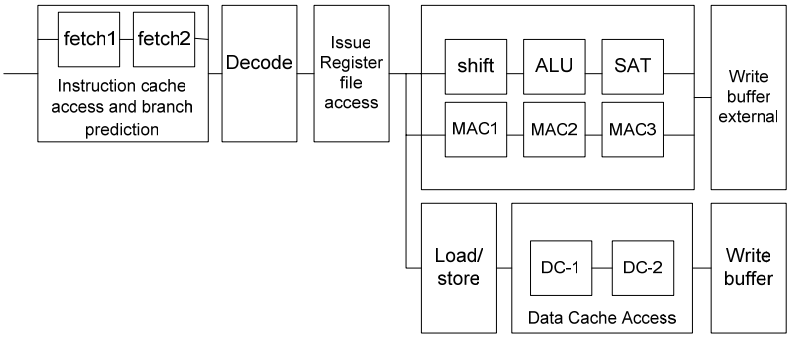

Figure 5. ARM11 processor pipeline

methodology used and then present results for different cache configurations.

5.1 Experimental methodology

The approach proposed in this paper was evaluated for several processor configurations shown in Table 2. A low-end processor uses 2, 4, 8 or 16KB instruction and data caches and no L2 cache. The high-end processor has two levels of on-chip caches, with L1 cache size of up to 32KB. The rest of the processor architecture is similar to the ARM11 family of processors [25].

Figure 5 shows the processor pipeline of ARM11, a single issue, out of order completion processor. It has two fetch stages and two stages for data cache access. The Fetch stages fill a four-entry instruction fetch buffer [25].

Table 2. Processors Configuration

	Low-end configuration	Medium-end configuration
L1 I-cache	2-4-8-16KB, 4 way, 2 cycles	4-8-16-32KB, 8 way 2 cycles
L1 D-cache	2-4-8-16KB, 4 way, 2 cycles	4-8-16-32KB, 8 way 2 cycles
L2-cache	none	64-128-256-512KB, 15 cycles
Fetch, dispatch	1 wide	1 wide
Issue	in-order, non blocking	in-order, non blocking
Memory	30 cycles	80 cycles
Instruction fetch queue	4	4
Load/store queue	4 entry	8 entry
Arithmetic unit	1 integer, 1 floating point units	1 integer, 1 floating point units
Complex unit	1 INT, 1 FP multiply/divide units	1 INT, 1 FP multiply/divide units
Pipeline	8 stages	8 stages
Processor speed	300 MHz	800 MHz

The processor supports non-blocking and hit-under-miss operations in which it continues execution after a cache miss, as long as subsequent instructions are not dependent on cache miss data. The processor pipeline stalls only after three successive data cache misses.

An extensively modified MASE [11] simulator was used to model the architecture. The MiBench suite [26] was used to represent a (low-end) embedded domain. SPEC2K benchmarks were used for the high-end processor. All benchmarks were compiled with the -O4 flag using the Compaq compiler targeting the Alpha 21264 processor. MiBench benchmarks executed for 500 Million instructions and SPEC2K for 500 Million instructions after fast-forwarding for 500 Millions instructions.

5.2 Reducing Leakage in L1 Data Cache

To maximize the leakage reduction in DL1 cache peripherals one simple solution is to always put them into ultra low power mode. However, this requires wakeup of DL1 peripheral circuits before cache access and adds 4 cycles to the DL1 latency which significantly reduces performance. One can put DL1 peripherals into the basic low power mode, which requires only one cycle to wakeup, and hide this latency during address computation stage, thus not degrading performance. However, this doesn't noticeably reduce leakage power (see Table 1). To benefit from large leakage reduction of ultra and aggressive low power modes and low performance impact of basic-lp mode one has to dynamically adjust the peripheral circuit sleep power mode such that during periods of frequent access they are kept in basic-lp mode and when they accessed infrequently they are put into aggr-lp or ultra-lp modes.

In this architecture it can be determined whether an instruction is load or a store at least one cycle before cache access (during issue stage in Figure 5). As a result, accessing DL1 while its peripherals are in basic-lp mode doesn't require an extra cycle because it's peripheral circuits can be woken up one cycle prior to access. Similarly, one cycle of wakeup delay can be hidden for all other low-power modes. Thus the effective wakeup overhead of DL1 cache is one cycle less than the delays shown in Table 1.

Based on the above, the DL1 is in basic-lp mode by default. For low-end processors the DL1 is accessed infrequently, if at all, once there is one or more pending DL1 cache misses. Thus it can be put into ultra low power mode. For high-end processors the DL1 is accessed very infrequently while an L2 cache miss is being

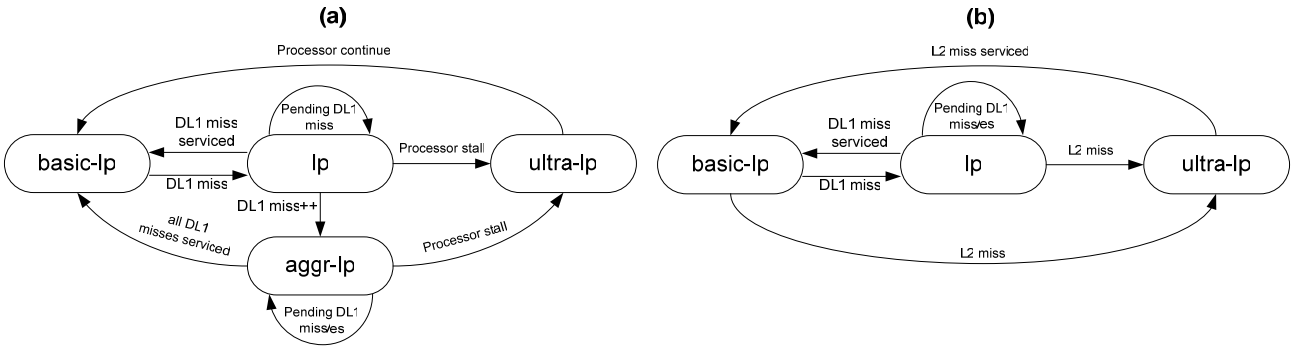

Figure 6. State machines to control DL1 cache peripherals in (a) low-end processor (b) medium-end processor

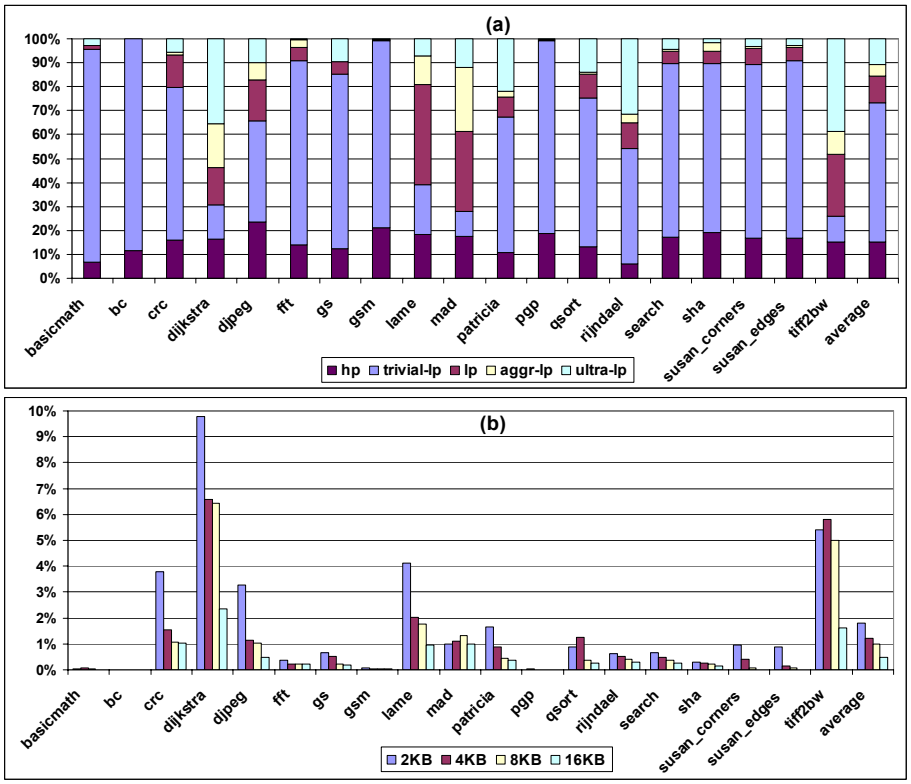

Figure 7. (a) Fraction of total execution time a 2KB DL1 cache spends in each of the power mode (b)Performance degradation of putting DL1 into low power mode (low-end architecture)

serviced. It is also access infrequently with no L2 cache misses but with multiple outstanding L1 misses. Based on these considerations the following state machines are proposed for different cache configurations.

5.2.1 Low-end architecture

For low-end architecture, a state machine shown in Figure 6(a) is proposed for controlling the DL1 low-power mode. Once a DL1 cache miss occurs the peripheral circuits transition from basic to lp mode. Given the miss service time of 30 cycles in this architecture, it is likely that processor stalls during the miss service period. Occurrence of additional cache misses while one DL1 cache miss is already pending further increases the chance of pipeline stall. Thus the DL1 peripherals are put into the aggressive low-power mode with more leakage savings.

Finally, the cache peripherals are put into the deepest low power mode, ultra-lp, once processor stalls after DL1 misses occurs. A stall is detected in the issue stage and the processor put into ultra-lp once the processor doesn't issue any instructions for at least five consecutive cycles after a DL1 miss.

Processor returns to the basic-lp mode from any of the other low power states when one of the two following conditions are met:

- Stall condition removed; i.e instruction issue resumes
- All pending DL1 misses are serviced

Figure 7(a) reports the fraction of total execution time a 2KB DL1 cache spends in each of the power modes for MiBench benchmarks. On average, 85% of the time DL1 cache peripherals can be put into one of the low power modes. Most of the time is spent in the basic-lp mode, 58% of total execution time. Figure 7(b) shows performance degradation for different DL1 cache

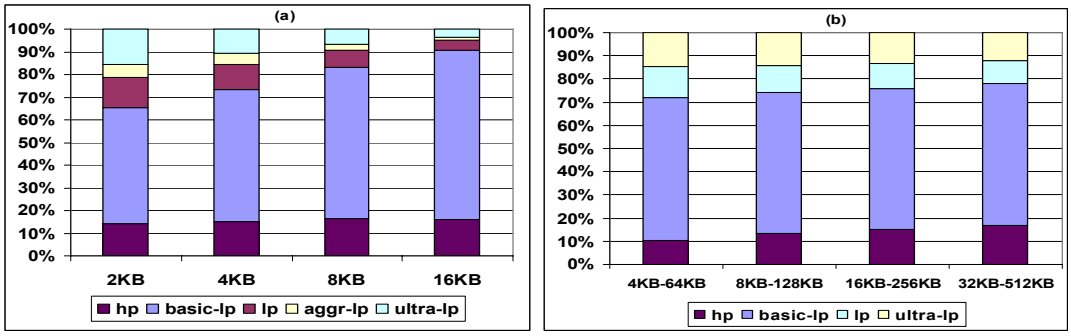

Figure 8. Fraction of total execution time DL1 is in different low power modes for (a) low-end and (b) high-end processors.

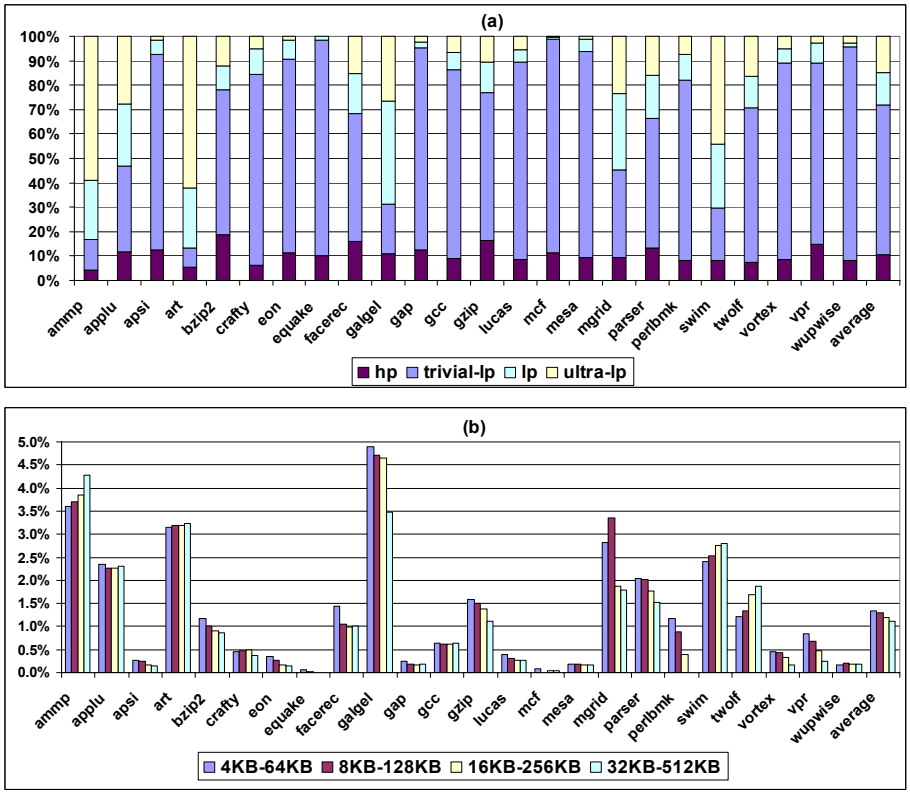

Figure 9. (a) Fraction of total execution time a 4KB DL1 cache spends in each of the power modes (b) Performance degradation of putting DL1 into low power mode (high-end architecture)

sizes. The average performance degradation is less than 2% in all cases. Interestingly, the benchmarks which spend a considerable amount of time in one of lp, aggr-lp or ultra-lp modes have the most performance degradation, for instance dijkstra, lame and tiff2bw. This is understandable as transition to these modes incurs larger time delay.

Figure 8(a) shows the fraction of total execution time a DL1 spends in different low power modes for different cache sizes. Increasing the cache size reduces DL1 cache miss rate and reduces opportunities to put the cache into more aggressive low power modes. This also reduces performance degradation for larger DL1 cache as can be seen in Figure 7(b).

5.2.2 High-end architecture

For high-end architecture we propose a simpler state machine to control the peripheral power mode (shown in Figure 6(b)). The major difference is that in this architecture the DL1 cache transitions to ultra-lp mode right after an L2 miss occurs. Given a long L2 cache miss service time (80 cycles) the processor will stall waiting for memory. The cache returns to the basic-lp mode once the L2 miss is serviced.

Figure 9(a) shows the fraction of total execution time a 4KB DL1 cache spends in each power mode for the high-end configuration. It shows that leakage power cannot be saved during only 10% of execution time. For the rest, the basic-lp mode has the highest contribution. Interestingly, in many benchmarks the ultra-lp mode

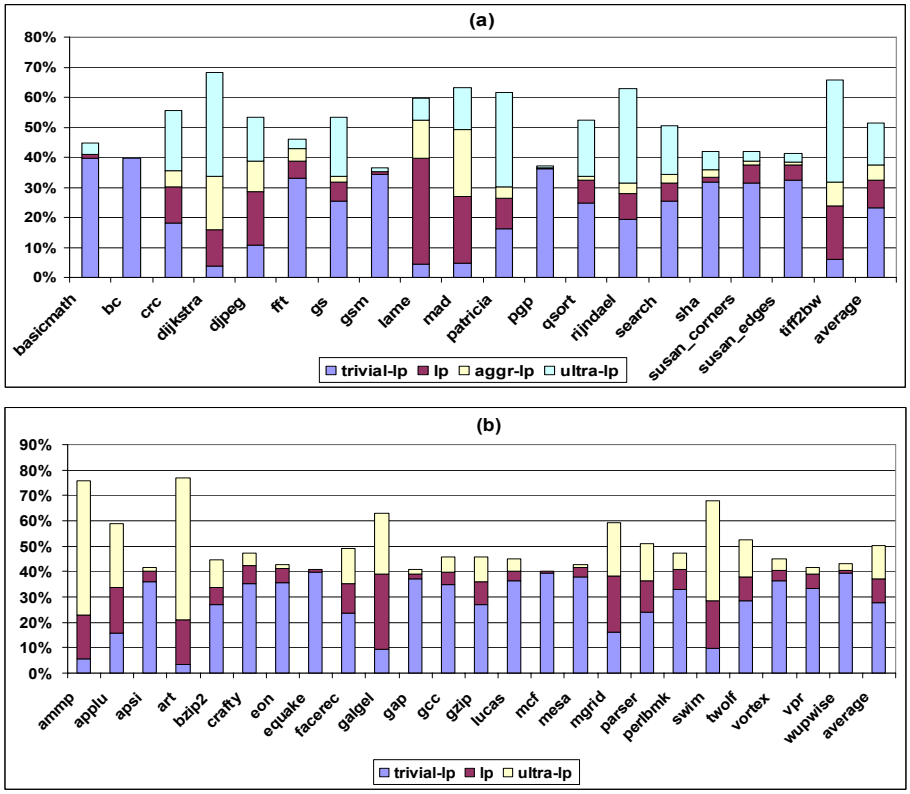

Figure 10. (a) Leakage power reduction for the low-end architecture with 2KB of DL1 cache (b) Leakage power reduction for the high-end architecture with 4KB DL1 cache

has a considerable contribution, e.g. for ammp, applu, art and swim. In fact, these benchmarks have high L2 miss rate (not shown here), which triggers transition to ultra low power mode. Figure 9(b) shows performance degradation for different DL1 cache sizes. The average performance degradation is less than 1.5%. Similar to low-end architecture, here the benchmarks which spend considerable amount of time in one of lp, aggr-lp or ultra-lp mode have the highest performance degradation: ammp, applu, art and galgel.

Figure 8(b) shows the fraction of total execution time DL1 is put into different low power modes for different cache sizes. Similar to low-end architecture, increasing the cache size in any of high-end configurations reduces the relative power savings period.

6. POWER AND ENERGY-DELAY RESULTS

This section presents results for power reduction and energy-delay product. First, let us describe power and timing assumptions used. We used the relative leakage power reduction of various power modes reported in Table 1. 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) reports the leakage power reduction of individual benchmarks for the low-end architecture with 2KB of DL1 cache. On average, DL1 leakage is reduced by 50%. The fraction of leakage power reduction of each of low power mode is also shown in the figure. Comparison of results in Figure 7 and Figure 10(a) shows that while ultra-lp mode occurs much less frequently compared to basic-lp mode, its leakage reduction is comparable to the basic-lp mode. The reason is that in ultra-lp mode the peripheral leakage is reduced by 90%, almost twice that of basic-lp mode.

Figure 10 (b) shows the leakage reduction results for the high-end architecture with 4KB of DL1 cache. The average leakage reduction is almost 50%. In Figure 11 (a) we report the leakage power and energy-delay reduction of different processor configurations and different DL1 cache size. On average, leakage is reduced by 42 to 52% for different configurations. In both low-end and high-end architectures larger DL1 caches have lower leakage savings. Overall, the energy-delay product reduction, unlike leakage power reduction, increases for larger cache size. The reason is that for smaller cache size, the fraction of dynamic-energy per access to static energy is noticeably higher. As a result, for these small caches a large leakage reduction does not translate to large overall energy-delay reduction. This is different for larger caches, as their static power dissipation is proportional to the dynamic power dissipation. The average energy-delay reduction varies from 9 to 21% for different architectures.

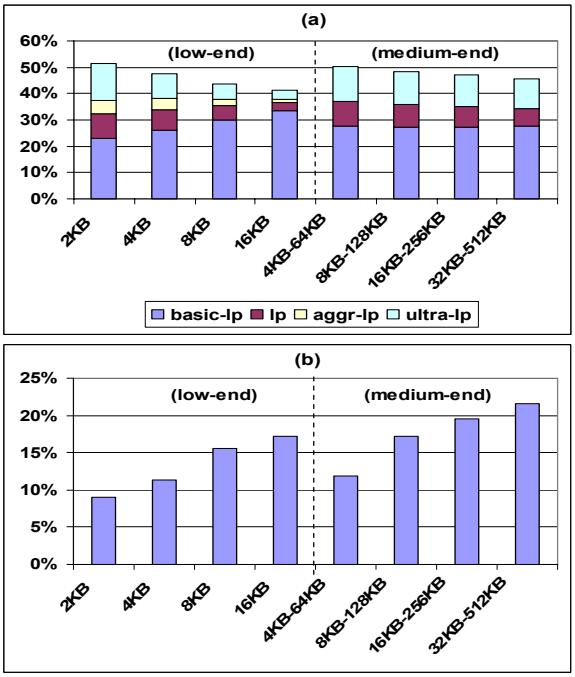

Figure 11. (a) Leakage power reduction (b) Total energy-delay reduction for DL1 cache.

7. CONCLUSIONS

This paper deals with the important problem of leakage energy in peripheral circuits of SRAM in L1 caches of embedded processors. These circuits account for 85% of overall leakage in 2 to 16KB caches in 65nm technology. By defining multiple sleep modes and architectural control of sleep mode transition, significant leakage energy reduction was achieved with no significant performance impact. In future technology of 45nm and below the peripheral leakage is expected to be even higher and therefore the proposed approach will result in even higher energy savings.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants NSF CCF–0311738 and CNS–0220069..

9. REFERENCES

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

[2] H. Homayoun and A. Veidenbaum, ZZ-HVS: Zig-Zag Horizontal and Vertical Sleep Transistor Sharing to Reduce Leakage Power in On-Chip SRAM Peripheral Circuits. Proc. IEEE ICCD, 2008.

[3] K.-S. Min et al., "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: an alternative to clock-gating scheme in leakage dominant era," ISSCC 2003.

[4] M. Horiguchi et al., "Switched-source-impedence CMOS circuit for low-standby subthreshold current giga-scale LSI's," Symp. VLSI circuits Dig. Tech. Papers, pp. 47-48, 1993.

[5] Y. Meng, T. Sherwood, and R. Kastner. On the limits of leakage power reduction in caches. In HPCA-11, 2005.

[6] W. Zhang and J. S. Hu. Compiler-directed instruction cache leakage optimization. In Proc. In MICRO-35, 2002.

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

[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] A Sub-lW to 2W Low-Power IA Processor for Mobile Internet Devices and Ultra-Mobile PCs in 45nm Hi-K Metal Gate CMOS, in ISSCC-2008.

[10] J. C. Park, V. J. Mooney III. Sleepy stack leakage reduction. IEEE Trans. VLSI Syst. 14(11): 1250-1263 (2006).

[11] SimpleScalar4 tutorial, 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, Apr. 2004.

[13] M.D. Powell et al,. Gated Vdd: 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..

[15] K. Agarwal, H. Deogun, D. Sylvester, K. Nowka. Power gating with multiple sleep modes. In ISQED 2006.

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

[17] M. Mamidipaka, K. S. Khouri, N. Dutt and M. S. Abadir, Analytical models for leakage power estimation of memory array structures. CODES+ISSS 2004.

[18] B 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] 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.

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

[22] .Thoziyoor, N. Muralimanohar, J. H. Ahn, and N P. Jouppi "CACTI 5.1 Technical Report" HP Laboratories, Palo Alto, April 2, 2008.

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

[24] H. Homayoun and A. Veidenbaum, Reducing Leakage Power in Peripheral Circuit of L2 Caches, In Proc. IEEE Intl.

Conference on Computer Design (ICCD 2007), Lake Tahoe, Oct. 2007.

[25] "ARM11 MPCore Processor Revision: r1p0" Technical Reference Manual , infocenter.arm.com/ help/ topic/ com.arm.doc.ddi0360e/DDI0360E_arm11_mpcore_r1p0_trm .pdf

[26] MiBench Version 1.0. http://www.eecs.umich.edu/mibench/.

[27] J. Montanaro, et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor", IJSSC, November 1996.

[28] J. C. Ku, S. Ozdemir, G. Memik and Y. Ismail,"Power Density Minimization for Highly-Associative Caches in Embedded Processors", GLSVLSI 2006.

[29] 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, 2007.

[30] S. Chun and K. Skadron, "On-Demand Solution to Minimize I-Cache Leakage Energy with Maintaining Performance", Transaction on Computers, 2008.

[31] 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 .

[32] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," DAC, June 1998.