Exploiting Locality to Improve Leakage Reduction in Embedded Drowsy I-Caches at Same Area/Speed

Massimo Alioto, Paolo Bennati, Roberto Giorgi
Department of Information Engineering
University of Siena
Siena, Italy
{malioto,bennati,giorgi}@dii.unisi.it

Abstract **In this paper, a technique to reduce the leakage power consumption in embedded drowsy instruction caches (I caches) is proposed. The technique is called "Improved Drowsy" (ID), and adopts a more efficient strategy than standard Drowsy Caches (DCs) to turn off unused cache lines, based on locality. The implementation of ID caches requires minor changes, and the area/speed overhead associated with the additional circuitry is insignificant. The proposed technique is assessed through circuit and cycle accurate simulations on an L1 instruction cache embedded in an ARM XScale processor based system in a 65 nm CMOS technology. Results show that this technique is able to reduce the leakage power by 69% on average. Leakage of DC is shown to be significantly lowered with the proposed ID approach, being DC leakage greater than that of ID by up to 53%, and 10 15% typically.**

I. INTRODUCTION

Leakage power consumption is well known to be a major issue in the design of nanometer VLSI circuits [1], [2]. Cache memories are particularly sensitive to leakage for various reasons. First, the leakage power represents a sizeable fraction of the power budget, due to the low activity rate of cells and peripheral circuits [3]. Secondly, the cache power consumption often represents a considerable portion of the overall chip power and area, especially in high-performance embedded processors [3], [4], [5]. Thirdly, at each new technology node, the leakage power is increasing faster than the dynamic power [1], [2]. For these reasons, new techniques to reduce leakage of caches are continuously required.

In the last decade, various techniques to reduce leakage in caches were devised at the boundary of circuit and architectural level [2]-[4]. Among these approaches, various techniques were proposed that dynamically vary the supply voltage of each cache line depending on whether it operates in active mode (normal operation, high leakage) or standby mode (not immediately accessible, low leakage). Two well-known techniques that exploit this mechanism are the Drowsy Cache (DC) [6] and Cache Decay (CD) schemes [7], both of which were demonstrated to significantly reduce leakage.

In this paper, a novel "Improved Drowsy" (ID) cache scheme to improve the Drowsy Cache technique is proposed. ID takes advantage of locality to implement a more efficient approach to turn cache lines into standby mode. Analysis shows that ID caches are able to reduce leakage with an insignificant speed and area penalty, compared to DC. Hence, the proposed scheme is a nice replacement of drowsy caches, as the leakage reduction comes at virtually no cost. ID are also shown to have a significant (2X) speed advantage compared to CD caches. Circuit and cycle-accurate simulations on a 65-nm technology are performed to assess the proposed technique.

REVIEW OF PREVIOUS TECHNIQUES

Caches are designed to exploit the *locality of references* in programs, both in space and time [8]. Locality is particularly evident in instruction caches (I-caches), where lines that are successively accessed are usually very close to each other (typically within 1-2 lines in most cases) [8], [10].

Let us consider the generic array structure of a cache memory in Fig. 1. Since only a minority of cells are accessed in a given time period, the majority of cells uselessly dissipate leakage power. To minimize the leakage associated with these non-accessed cells, they are usually grouped in cache lines that can operate in active or standby mode [9]. In active mode, a cache line is powered with the nominal supply voltage V_{DD}, and operates normally. In standby mode, the cache line voltage is reduced to a lower value $V_{DD,L}$ that reduces the leakage, but it does not allow an immediate access to the corresponding cells. The operating voltage of a cache line is set by the corresponding cache line controller (CLC), as in Fig. 1. In the case of drowsy caches (DC), $V_{DD,L}$ is set to a low value (typically 0.3-0.4 V) that still allows the cell to retain the previously stored data [6]. Hence, when a cache line in standby mode is accessed, its voltage must be first switched from $V_{DD,L}$ to V_{DD}, which usually requires one additional cycle and an additional dynamic energy contribution. Hence, DC pays a small speed and dynamic energy overhead, compared to caches with no cache line voltage control [6].

In the simplest Drowsy Cache scheme, the global signal *all drowsy* periodically sets all cache lines in standby mode at

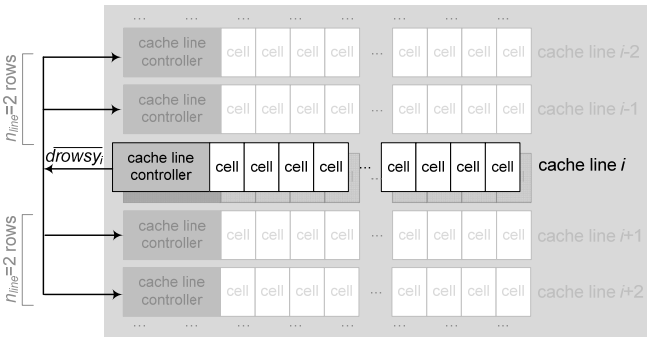

Fig. 1. Cache array and routing of signal $\overline{drowsy_i}$ to n_{line} 2 cache lines above and below the *i*-th line (other signals are omitted).

the beginning of a time period t_{DROWSY} called "update window" (usually a few thousands of clock cycles) [6]. Then, the accessed cache lines are set in active mode and stay in the same mode until the end of the update window. In detail, the operation mode of a cache line is set by an SR Flip-Flop (FF) included in the corresponding cache line controller in Fig. 1): if the stored signal $drowsy_i$ is 1 (0), the *i*-th line is in standby (active) mode. When the *all drowsy* signal turns to high at the beginning of the update window, input *R* of all FFs is set to 1 and hence the cache lines are all set to standby mode.

Cache decay (CD) approach is not very different from DC, except for the fact that $V_{DD,L}$=0 V, which permits to further reduce the leakage power compared to DC [7]. However, since $V_{DD,L}$=0 V, cells in standby mode are not able to retain the previously stored information, and a cache miss occurs whenever a "decayed" cache line in standby mode is accessed (hence, a high latency must be paid to wait for the recovery of the missed data from the slower RAM memory). For these reasons, CD caches exhibit a lower leakage compared to DC, but have a considerably worse performance (an access to a decayed line is nothing but a miss).

REDUCING LEAKAGE IN DROWSY I-CACHES

The main limit of DC in reducing leakage is that a cache line activated at a given point of time is kept active until the completion of the update window t_{DROWSY}. Hence, even if a cache line is used just once within the update window, it stays in active mode for the rest of the window, thus wasting power.

Leakage in Drowsy Caches may be further reduced by adopting an appropriate "Improved Drowsy" (ID) cache mechanism that turns active cache lines into standby mode immediately after they are not accessed any longer. This ID mechanism should be "simple" to minimize the power/area overhead (which may easily nullify the leakage reduction). Moreover, this ID mechanism should be "local", in the sense that it must involve a small number of cache lines. Indeed, the circuit implementation of a centralized controller monitoring the activity of all cache lines is unfeasible, since it would require a complex and power-hungry routing to all cache lines.

In the following, locality of I-caches is exploited to devise a novel technique to reduce leakage that satisfies both requirements (simple, local). It is well known that I-caches exhibit high spatial locality, i.e. newly accessed cache lines are physically close to the lines that were accessed previously [8], [10]. Hence, when a new cache line is accessed, it is likely that some nearby line was previously used, but it will not be used any longer. Hence, in most cases the previously active lines that have to be turned again in standby mode must be searched in the neighborhood of the currently accessed cache line. This consideration enables a "local" mechanism to turn into standby mode the cache lines that are no longer used. Indeed, when the generic *i*-th cache line is accessed, the high value of its drowsy signal $\overline{drowsy_i}$ indicates that this line is in active mode. Hence, signal $\overline{drowsy_i}$ can be sent directly to the nearby cache lines to force them to turn into standby mode, without the need of a centralized control circuitry. More specifically, signal $\overline{drowsy_i}$ is sent to n_{line} cache lines before and after the *i*-th line (i.e., to adjacent cache lines with index ranging from i-n_{line} to i+n_{line}, excluding the *i*-th line).

According to discussion in Section II, we will reasonably assume n_{line}=2, hence the signal $\overline{drowsy_i}$ must be distributed only to the two lines above and below the *i*-th line, as shown in Fig. 1. Hence, the wires driven by $\overline{drowsy_i}$ are very short, and the resulting area and power overhead is expected to be very small. Once signal $\overline{drowsy_i}$ of each cache line is distributed to the $2n_{line}$ adjacent lines as in Fig. 2a, the drowsy cache line controller (CLC) of the generic *i*-th line must be slightly modified w r.t. Drowsy Caches. Indeed, CLC must force the *i*-th line into standby mode only if the following conditions are satisfied at the same time: 1) the considered *i*-th line is not selected by the row decoder (i.e., if sel_i=0, being sel_i the cache line selection signal provided by the row decoder), 2) if some nearby line is in active mode (i.e., if $drowsy_j$=0 for some $j \neq i$ ranging from i-n_{line} to i+n_{line}) or if the signal *all drowsy* that triggers the new update window is set to 1 (according to the conventional drowsy scheme). Accordingly, since the standby mode of a cache line is forced by setting *R*=1 in the SR Flip-Flop in Fig. 2a storing the cache line state [6] (drowsy/active), signal R_i in the CLC of *i*-th cache line has the following logic expression:

$$R_i = \left(\sum_{\substack{j=i-n_{line} \\ j \neq i}}^{i+n_{line}} \overline{drowsy_j} + all\_drowsy \right) \cdot sel_i \qquad (1)$$

where symbol Σ is the logical sum (i.e., OR) of terms $\overline{drowsy_j}$ associated with adjacent lines that are far at most n_{line} rows.

From the above considerations, the cache line controller of the proposed ID caches is very similar to that of drowsy caches [6], where the drowsy state logic (which generates R_i) is replaced by the block in Fig. 2a. Actually, a more compact implementation of (1) can be achieved by embedding the logic function in (1) into the SR Flip-Flop as shown in Fig. 2b, and implementing the Flip-Flop in the well-known 6T topology [6] (see transistors M1-M6 in Fig. 2b). In the resulting transistor-level implementation reported in Fig. 2b, the cache line controller has only seven transistors more than the conventional Drowsy Cache controller [6] (i.e., M7-M11 and the two transistors of the inverter driving transistor M11).

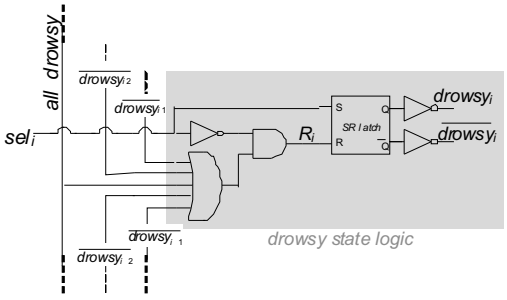

Fig. 2a. ID cache gate-level implementation of (1) in the drowsy state logic.

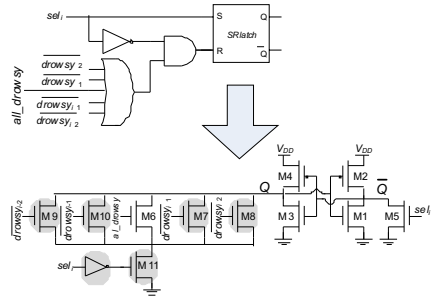

Fig. 2b. Efficient circuit implementation of drowsy state logic in ID caches.

These transistors can be minimum sized since the cache line controller is not required to be fast [6].

DESIGN CONSIDERATIONS ON THE ID TECHNIQUE

In this section, practical issues are discussed. First, it should be observed that the *all drowsy* signal in (1) used in conventional drowsy caches is still needed in the proposed scheme. Indeed, if the signal *all drowsy* was not available, a given line accessed for one time might remain in active mode indefinitely, if the adjacent lines are in standby mode[1]. For this reason, a large number of lines may remain in active mode even if they are no longer accessed, thus severely increasing the leakage power consumption. Therefore, to make sure that unused lines are periodically turned into drowsy, and hence to avoid the above leakage power waste, the update window technique used in conventional drowsy caches is still used in the proposed approach.

It is worth noting that the proposed technique is less effective in reducing leakage for code that contains many jumps. However, program exhibits sequential locality to a great extent [9], [10], hence it is expected that the proposed technique provides an appreciable leakage reduction in most practical cases. In regard to the speed, both conventional and improved drowsy caches incur in a one-cycle penalty when lines in drowsy state are accessed. Scarce sequential locality may affect the proposed technique also on the performance side. Anyway, we experimentally found that such influence is very limited, as discussed in Section V.

Finally, observe that there is virtually no difference between DC and ID caches in terms of cycle time and impact of process/voltage/temperature variations, as the proposed ID cache scheme differs by DC only by the seven transistors in Fig. 2b (which affect speed - and its variations - very weakly).

SIMULATIONS, RESULTS AND REMARKS

The benefits and drawbacks of the proposed technique were evaluated by considering L1 I-caches with size ranging from 8 KB to 32 KB and operating at 400 MHz, which are typical values in embedded caches for low-power systems [11]. The considered I-caches are separated from the data cache, are direct-mapped and have a 64-B cache line size, 1-bank organization, single read/write port, segmented bitlines and wordlines, H-tree organization for address/data bus interconnects (no L2 cache was used). We adopted the typical 4096-cycle update window [9] and extensively used clock gating. The caches were accessed by an ARM Xscale processor, on which all MiBench benchmarks run [12]. MiBench is a free and general suite widely accepted as a representative set for embedded systems [12].

The caches were simulated by resorting to mixed circuit and cycle-accurate simulations, as it is usually done in the analysis of cache memories [4], [7], [11]. Spectre simulations were performed to extract the main power and delay contributions of each block associated with cache accesses. Wire capacitances were extracted by assuming a 6T cell size of 1.3 μm X 0.6 μm. A 65-nm design kit was used to perform these simulations and extract wire parasitic, and V_{DD} was assumed to be 1.1 V ($V_{DD,L}$ in standby mode was set to 0.3 V). Data resulting from the circuit simulations were then used to perform cycle-accurate simulations with HotLeakage simulator [13] (version for ARM-based processors), which was modified to implement the proposed ID cache technique. The CACTI 5.3 simulator was included in the HotLeakage source code to estimate access time, energy and area for the considered cache parameter values [14].

The proposed Improved Drowsy technique (ID) was compared to the Drowsy Cache (DC) and Cache Decay (CD) approaches. The average leakage reduction factor of the three techniques with respect to the case with none of these techniques is reported in Table I, where data are averaged across all MiBench benchmarks and cache sizes. To perform a fair comparison, in Table I the power overhead paid for the implementation of these techniques (i.e., the power of the cache line controller and the power dissipated to switch the cache lines) was included in the leakage evaluation.

As expected, CD offers the greatest leakage reduction (4.09X), but pays for a dramatic speed penalty (IPC is reduced by a factor of about 2), which makes CD not competitive for moderate- to high-performance applications. On the other hand, DC has negligible performance degradation (IPC increases by a factor of only 1.003), and still permits a significant leakage reduction (although lower than CD).

The proposed ID technique was confirmed to be effective in reducing leakage. For example, this is shown in Fig. 3, where the leakage reduction in an 8-KB cache is plotted for each benchmark (very similar results were obtained for the other cache sizes). From this plot, the leakage reduction permitted by ID caches is rather consistent among the benchmarks, and is 69% on average.

Interestingly, the above leakage reduction of ID caches comes at negligible speed penalty. Indeed, from the plot of IPC reduction of ID with respect to a standard cache without any low-leakage technique in Fig. 4, the IPC is almost unaffected by the ID technique. More specifically, the ID technique determines an IPC degradation that is typically less than 1% (according to Table I), slightly greater than 1% for a couple of benchmarks, and significantly greater (11%) only for the benchmark *bitcount*. Accordingly, virtually no speed degradation is observed in ID caches.

When ID is compared to DC, it exhibits a better leakage reduction by keeping essentially the same area, performance and dynamic power. In detail, the resulting leakage increase of DC with respect to the proposed ID cache in the adopted 65-nm technology is shown in Fig. 5 for an 8- KB cache and various benchmarks (very similar results were obtained for

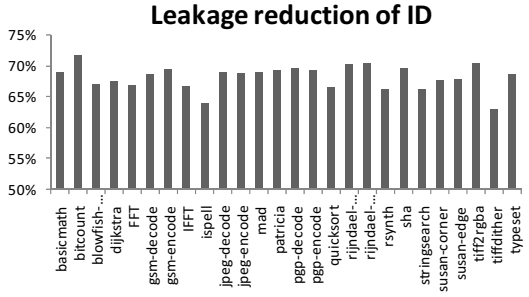

Fig. 3. Leakage reduction in a 8KB ID I-cache vs. MiBench test.

[1] This is clear from (1) by setting *all drowsy*=0 Indeed, if the adjacent lines are in drowsy mode (i e , $\overline{drowsy_j}$=0 in (1)), R_i remains at 0 and thus the i-th line remains in active mode

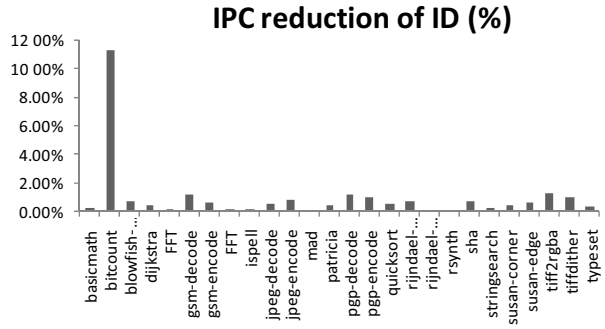

Fig. 4. IPC reduction (lower is better) for a 8KB i-cache. ID vs DC.

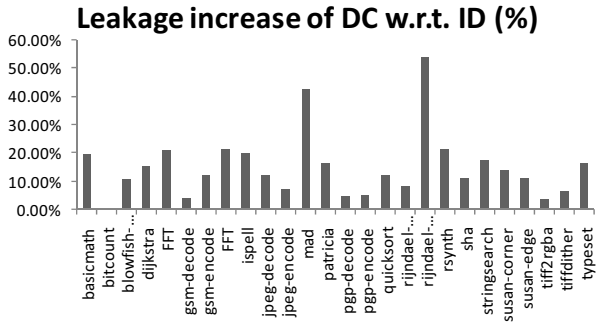

Fig. 5. Leakage increase of DC with respect to the proposed ID cache (higher is better) for a 8KB I-cache: ID vs DC.

other cache sizes). From Fig. 5, Drowsy Caches have a significantly higher leakage consumption compared to ID caches. Indeed, leakage of DC is higher than ID typically by 10-15%, and up to 53% (for the *rijndael encode* benchmark). As expected from the considerations in Sections III-IV, this is because ID caches are able to turn a greater number of cache lines in standby mode, compared to DC. Note that ID caches have almost the same leakage as DC (within 1%) in a few benchmarks (*tiff2bw*, *CRC32*, *tiffmedian*, *susan smooth*, *ADPCM encode*, *ADPCM decode*, *bitcount*).

ID caches also have an insignificant area overhead compared to DC. Indeed, the only circuit overhead consists of seven minimum-sized transistors (highlighted in gray in Fig. 2b) for each cache line (which, on the other hand, consists of thousands of transistors). In regard to the performance, the average IPC of ID caches was found to be very close to DC (within less than 1%, as shown in Table I). Again, this confirms that ID caches have a negligible speed degradation, as previously discussed. Regarding the dynamic power consumption, it was also found to be the same as DC. Indeed, any power overhead associated with the implementation of the technique was included in the evaluation of leakage, as pointed out above. According to these results, the proposed ID technique is equivalent to DC in terms of performance, area and dynamic power, but it further reduces leakage.

In regard to the comparison with CD, as expected the latter has a lower leakage power compared to the other techniques, but its performance is severely worse (see Table I). More in detail, the IPC of ID is on average 2X (and up to 8X for some benchmarks) better than that of CD. This is because the CD mechanism pays a large speed penalty due to the frequent misses associated with the access to standby lines. Again, this confirms that CD is not suitable for moderate- to high-performance applications, and its increased average leakage reduction (better than ID by 27% from Table I) does not justify its adoption.

TABLE I. COMPARISON OF DIFFERENT SCHEMES ACROSS MIBENCH SUITE FOR THE BASELINE CONFIGURATION.

	CD	DC	ID
leakage reduction factor	4.09X	2.7X	3.2X
IPC degradation factor	1.98X	1.003X	1.01X

CONCLUSION

In this paper, the Improved Drowsy cache technique was proposed as a more efficient scheme to turn unused cache line into standby mode. Analysis has showed that the proposed technique is a good replacement of Drowsy caches, as it permits to reduce leakage (typically by 10-15%, up to 53%) at virtually no cost in terms of area/speed/dynamic power. This is enabled by the more efficient mechanism according to which cache lines are turned on and off, which explicitly relies on locality.

ACKNOWLEDGEMENTS

This work was partially funded on EU projects HiPEAC (IST-217069), ERA (249059), TERAFLUX (249013).

REFERENCES

[1] S. G. Narendra, A. Chandrakasan, *Leakage in Nanometer CMOS Technologies*, Springer, 2006.

[2] Y. Nakagome, M. Horiguchi, T. Kawahara, K. Itoh, "Review and future prospects of low-voltage RAM circuits", in *IBM J. R&D*, vol. 47, pp. 525-552, 2003.

[3] K. Itoh, M. Horiguchi, H. Tanaka, *Ultra-low Voltage Nano-scale Memories*, Springer, 2007.

[4] N. S. Kim, D. Blaauw, T. Mudge, "Quantitative analysis and optimization techniques for on-chip cache leakage power", in *IEEE Trans. on VLSI Systems*, vol. 13, 2005, pp. 1147-1156.

[5] C. H. Kim, J. J. Kim, S. Mukhopadhyay, K. Roy, "A forward body-biased low-leakage SRAM cache: device and architecture considerations," in *Proc. of ISLPED 2003*, pp. 6-9, 2003.

[6] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, T. Mudge, "Drowsy caches: simple techniques for reducing leakage power", in *Proc. of ISCA'02*, pp. 148-157, 2002.

[7] S. Kaxiras, Z. Hu, M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power", in *Proc. of ISCA'01*, pp. 240-251, 2001.

[8] D. A. Patterson, J. L. Hennessy, *Computer organization and design the hardware/software interface* (2nd ed), Morgan Kaufmann, 1997.

[9] D. Parikh, Y. Zhang, K. Sankaranarayanan, K. Skadron, and M. Stan, "Comparison of State-Preserving vs. Non-State-Preserving Leakage Control in Caches", in *Workshop on Duplicating, Deconstructing and Debunking* (held in conjunction with ISCA'03), pp. 14 25, 2003.

[10] R. Giorgi, C. A. Prete, and G. Prina, "Cache Memory Design for Embedded Systems Based on Program Locality Analysis", in Proc. of MSE'97, pp. 16-18, 1997.

[11] Intel, "The Intel XScale Microarchitecture", in Technical Summary - av. at: *http //www.intel.com/design/intelxscale/xscaledatasheet4.htm, 2000*.

[12] M. R. a. R. Guthaus, J.S. and Ernst, D. and Austin, T.M. and Mudge, T. and Brown, R.B., "MiBench: A free, commercially representative embedded benchmark suite", *in Proc. of* WWC'01, pp. 83-94, 2001.

[13] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects", University of Virginia Tech report, Charlottesville CS-2003-05, 2003

[14] S. Thoziyoor, N. Muralimanohar, J. Ahn, and N. P. Jouppi, "CACTI 5.1", Technical Report HPL-2008-20, HP Labs