Taking Advantage of Within-Die Delay-Variation to Reduce Cache Leakage Power Using Additional Cache-Ways

Goudarzi, Maziar
System LSI Research Center, Kyushu University

Matsumura, Tadayuki
Graduate School of Information Science and Electrical Engineering, Kyushu University

Ishihara, Tohru
System LSI Research Center, Kyushu University

出版情報：4th International Workshop on Dependable Embedded Systems, 2007-10-09
バージョン：
権利関係：

Taking Advantage of Within-Die Delay-Variation to Reduce Cache Leakage Power Using Additional Cache-Ways

Maziar Goudarzi†, Tadayuki Matsumura‡, Tohru Ishihara†
{†System LSI Research Center, ‡Graduate School of Information Science and Electrical Engineering}, Kyushu University, Fukuoka, Japan
{goudarzi, ishihara}@slrc.kyushu-u.ac.jp matsumura@c.csce.kyushu-u.ac.jp

Abstract

Leakage power, especially in cache memories, is dominating total power consumption of processor-based embedded systems. By choosing a higher threshold voltage, SRAM leakage can be exponentially reduced in return for lower speed. Since SRAM cells in the same cache have different delays in nanometer technologies due to within-die process variation, not all of the cells violate the cache delay. However, since timing-violating cells are randomly distributed over the cache, row/column redundancies are inefficient. We propose to add extra cache-way(s) to replace slow cache-lines separately in each cache-set. In a commercial 90nm process, our technique can reduce leakage power by up to 54% which, depending on the share of leakage in total cache power, translates to 25.36% and 53.37% reduction of total energy respectively in L1 and L2 cache by adding two spare ways to a 4-way set-associative cache with no performance penalty.

1 Introduction

The share of leakage in total power consumption of cache memories increases with every new technology node. In addition, leakage exponentially increases with temperature. The naïve solution to exponentially reduce leakage power is to use higher threshold voltage (V_{th}) and/or gate-oxide thickness (T_{ox}), but this slows down the SRAM cells speed. Another effect, which is increasingly more pronounced in sub-90nm processes, is random within-die variation in SRAM delay; these variations generally follow Gaussian distribution [1]. In the absence of such variations, all SRAM cells have the same delay which also defines the cache delay. But in presence of random within-die delay variation, the cache delay has to be farther from the mean delay of SRAM cells so as to obtain a reasonable timing-yield for the cache-containing chip. In such case, at higher V_{th} and/or T_{ox}, only a subset of the cells (which are randomly distributed over the cache) violate the target cache delay; we use extra cache-ways to compensate for them.

Several previous works address improving cache-memory timing-yield in presence of process variation by proposing process-tolerant cache architectures [1], [2] and code-placement compiler techniques [3], but they actually reduce the useful capacity of the cache by marking and avoiding to use too-slow cache lines. Although [3] provides a solution to mitigate the performance impact, it demands a per-chip different binary executable. Several other researches address cache power reduction by reducing dynamic power [4], [5] or static power [6], [7], but these do not consider process-variation effects and the corresponding yield loss.

In this paper, we propose an optimization technique for cache-design that is applied at design and manufacturing time of the cache-containing chip and reduces total power, by significantly reducing the leakage component, at the cost of extra chip area for additional cache ways. We propose *(i)* to keep V_{DD} untouched (so as not to impact dynamic power), *(ii)* to use a higher V_{th} and gate-oxide thickness (T_{ox}) (so as to exponentially reduce subthreshold leakage as well as gate leakage), and finally *(iii)* to add a few extra cache ways to compensate for the delay-violating cache-lines (so as to keep the original performance). We choose the number of extra cache ways and the value for V_{th} and T_{ox} such that leakage power is minimized while the cache capacity, speed, and timing-yield are all kept unchanged.

A major issue is the random spreading of delay-violating SRAM cells in the cache, which makes it inefficient to use row/column redundancy. Our technique answers this problem since we replace slow cache lines separately in each cache-set.

The rest of this paper is organized as follows. Section 2 reviews related works and presents our approach. In Section 3, the optimization problem is formulated and an algorithm is provided for that. Section 4 provides the experimental results and finally Section 5 summarizes and concludes the paper.

2 Related Works and our Approach

2.1 Related Works

Turning off unused parts of the cache [9] [6] or putting them in a low-energy "drowsy" mode using two different supply voltages [7] reduce leakage, but they require separate supply (V_{DD}) for each cache-line or ground lines to cutoff or reduce supply voltage, which have proven expensive. Our technique does not need any change in the cache line design and only replicates cache-ways and chooses manufacturing-time V_{th} and T_{ox} options available in most fabrication processes today.

Reverse body biasing [10] or increasing the V_{th} at manufacturing time can effectively reduce leakage, but this increases cells delay and results in lower performance and/or reduced timing-yield. Forward body biasing during active-mode [11] and dynamic V_{th} control [12] can improve delay, but naturally they increase leakage in the active mode. This delay impact can be compensated by increasing V_{DD} in line with the increase in V_{th}, but this quadratically increases the dynamic power [8]. We keep the original V_{DD} although we increase V_{th}; instead, we add extra cache ways to compensate for the cache ways that violate target delay due to the

increased V_{th}. Using spares to repair manufacturing defects [13] is not new, but to the best of our knowledge it has not been used to reduce leakage in the past.

A conventional technique to reduce leakage power in memories without performance/timing-yield penalty is supply-voltage scaling along with V_{th} scaling. In this approach, V_{DD} of the cache is increased, after the raise in V_{th} for leakage reduction, so that SRAM cells speed and the cache timing-yield are restored to the original values. The disadvantage, however, is the quadratic increase in dynamic power. To mitigate this disadvantage, various techniques exist that reduce dynamic power of cache [5]. A well-known technique applicable to instruction cache is [14][15] to substantially reduce dynamic power in set-associative instruction caches; since instructions are mostly executed sequentially, and several instructions usually reside in the same cache-line, tag-comparisons can be eliminated and only one cache way can be activated except when the last executed instruction either was a branch or was resided at the end of a cache-line. We call this technique *Inter-Line Way Memorization (ILWM)* and use it in our experiments in Section 4.

2.2 Motivational Example

Figure 1-a shows a sample 2-way set-associative cache with 4 sets. After uniformly increasing the V_{th} and/or T_{ox} of the cache SRAM cells to save leakage, the delay of all cells increases; however, since their individual delays were not the same at the beginning (due to the within-die delay variation) only some of them would now violate the original cache delay (the corresponding cache lines are painted in red in Figure 1-b). To compensate for such delay-violating cache-lines, the *Additional* cache way(s) is added so that still the same 2 delay-meeting cache-lines are available in all sets.

2.3 Analytical Example

We define the following notations:

μ_d: the original mean delay of SRAM cells.
σ_d: the original standard deviation of delay of SRAM cells.
***D*:** the target delay of the cache.
***N*:** the number of additional cache ways.
***Y*:** the original timing-yield of the cache.
Y_{cell}: the timing-yield of a single SRAM cell.
$Y_{set,N}$: timing-yield of a cache-set with *N* additional ways; the cache-set is still fault-free if at most *N* ways violate the timing.

Assume that a 4-way set-associative cache with 128-bits per cache-line and 256 cache-sets is to be used in an embedded system. Due to process variation, different SRAM cells of the cache will have different delays even in the same chip; this distribution is believed to follow Gaussian distribution [1]. The probability, Y_{cell}, that a single SRAM cell meets the *target delay, D,* can be thus given by equation (1) below:

$$Y_{cell} = \Pr[x \le D] = \int_{-\infty}^{D} f(x)dx \quad f(x) = \frac{1}{\sigma_d\sqrt{2\pi}} e^{\frac{-(x-\mu_d)^2}{2\sigma_d^2}} \qquad (1)$$

where *f(x)* is the Probability Density Function (PDF) of Gaussian distribution and μ_d and σ_d are respectively mean and standard deviation of the delay distribution. The probability of all cells in a cache-line meeting the target delay is

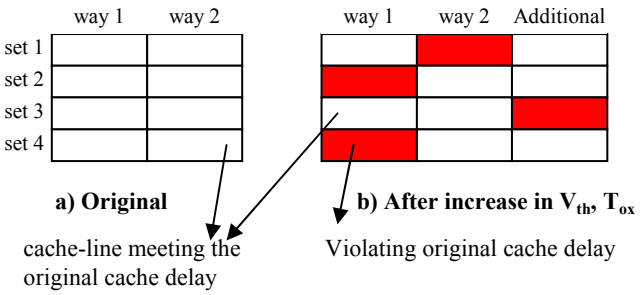

Figure 1. a) Original cache, b) after applying our technique

$Y_{line}=(Y_{cell})^{128}$ and for a complete cache-set the corresponding probability is $Y_{set,0} = (Y_{line})^4$ since this is a 4-way cache.

Now, if we add one extra way to the cache, each set would be working at least as good as before if at least 4 ways out of the available 5 ways meet the target delay. Thus:

$$Y_{set,1} = Y_{line}^{\ 5} + Y_{line}^{\ 4} \times (1 - Y_{line}) \times 5 \qquad (2)$$

Similarly, if there are two extra cache-ways, the probability $Y_{set,2}$ that the cache would still be as fine as before is even higher and is given by the following formula:

$$Y_{set,2} = Y_{line}^{\ 6} + Y_{line}^{\ 5} \times (1 - Y_{line}) \times 6 + Y_{line}^{\ 4} \times (1 - Y_{line})^2 \times 15 \qquad (3)$$

The timing-yield of the entire cache would then be $(Y_{set,N})^{256}$ where *N* represents the number of extra cache ways. Figure 2 depicts $Y_{set,0}$, $Y_{set,1}$, $Y_{set,2}$ for $0.999 \le Y_{cell} \le 1$ (for presentational purposes, we have depicted Y_{set} values instead of the yield of the entire cache in each case). For a given target timing-yield, the caches with extra ways demand a lower Y_{cell} (e.g. in Figure 2, for 97% target Y_{set}, the original cache requires Y_{cell} to be 99.996%, while with one extra way this reduces to 99.958%, and with two extra ways, it further reduces to 99.9%); lower Y_{cell} corresponds to higher delay (Figure 3) and lower power (Figure 4); this is detailed below.

Y_{cell} corresponds to the area below the PDF of the delay of a SRAM cell up to the upper bound *D* (gray area in Figure 3). Thus, a reduced Y_{cell} requirement means the PDF diagram can be shifted to the right by increasing μ_d and keeping the same *D*; this is illustrated in Figure 3. For example in Figure 2, for

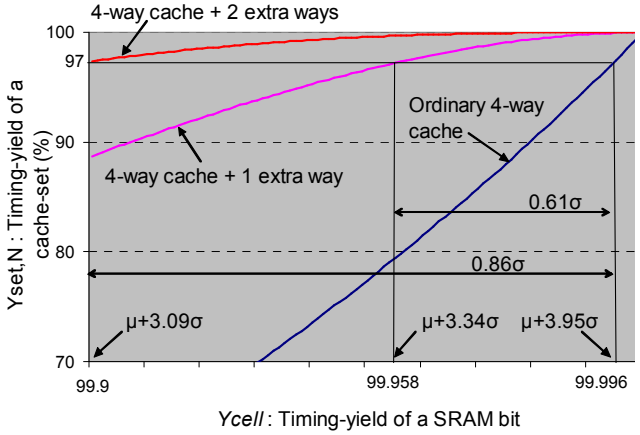

Figure 2. Timing-yield diagrams of ordinary, 1-extra, and 2-extra caches reflecting the available space for the increase in the mean-delay of cells.

the same timing-yield, the ordinary cache needs $D=\mu_d+3.95\sigma_d$ while with one additional cache way $D=\mu_d+3.34\sigma_d$ and for two additional ways $D=\mu_d+3.09\sigma_d$, which thus means μ_d can be increased by $0.61\sigma_d$ and $0.86\sigma_d$ respectively if D is desired to remain constant. Raising μ_d can be realized by increasing the V_{th} and T_{ox} of the transistors comprising the SRAM cells so that leakage (both subthreshold and gate leakage) are effectively reduced. This is shown in Figure 4 which gives the SPICE simulation results of a 16KB cache in a commercial 90nm process; the delay and leakage values are obtained for varying values of V_{th} and T_{ox}. The SPICE models are manufacturer-supplied and correspond to a middle-performance 90nm process technology; the leakage values correspond to a single SRAM cell.

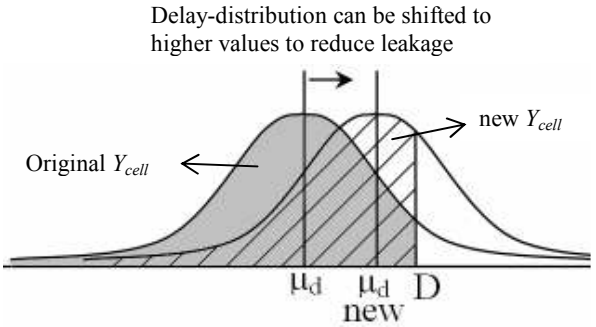

Figure 3. Lower Y_{cell} translates to higher mean-delay of SRAM cells resulting in lower leakage.

In summary, we propose that by adding extra cache ways, the leakage power of the cache memory can be effectively reduced, by changing their V_{th} and T_{ox}, without compromising performance or timing-yield or cache-capacity while also tolerating process variation. We choose the number of extra ways and the V_{th} and T_{ox} values such that timing-yield and target delay remain invariant. Adding extra cache ways, however, increases the dynamic power per access, and hence, the total power does not necessarily decrease. Obviously, the higher the share of leakage in the total cache power, the more effective this technique. We provide experimental results in Section 4 that reflect its effectiveness in real-life applications.

It is important to note that our technique is enabled by the within-die variation (higher σ_d gives more savings). Thus, this is a variation-based leakage reduction technique; *it saves leakage by choosing higher V_{th} and/or T_{ox}*, but the *within-die delay variation is essential* to restore the cache capacity, speed, and timing-yield afterwards by additional cache ways.

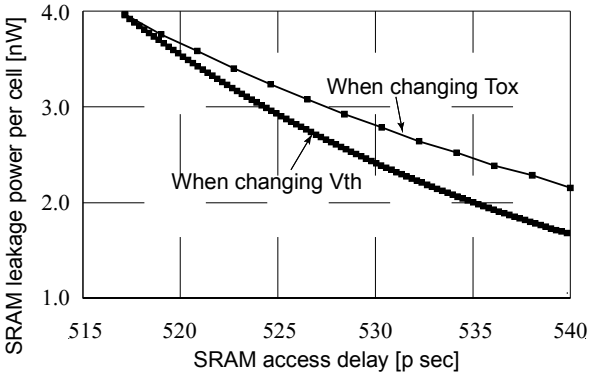

Figure 4. Leakage power vs. access-delay of a single SRAM cell when raising V_{th} and T_{ox} (from left to right) in a 90nm process technology.

2.4 Our Approach

Figure 5 shows the outline of our proposed approach. We propose a design- and manufacturing-time optimization that determines the number of extra cache-ways and the manufacturing options of T_{ox} and V_{th} of the transistors of the cache SRAM cells. The original cache organization (i.e. total size, line-size, and the number of ways) along with the process-technology characteristics (i.e., mean and standard-deviation of SRAM cell delay caused by within-die variation as well as leakage-delay curves of the cells at various V_{th} and T_{ox} values) are the inputs of the optimization program. The cache organization is modified according to the optimization results and the manufacturing option of T_{ox} and V_{th} is handed over to the manufacturer for chip fabrication. The produced chips are then tested offline to detect and mark cache lines containing slow SRAM cells. If the number of such slow cache lines exceeds the number of extra cache ways in each cache-set, the chip is considered faulty and contributes to loss of yield. It would have been best to cutoff the power-line of such slow cache lines to prevent them from leaking, but since this may not be practical, we rely on marking them at boot-time while their location is read from an agreed-upon non-volatile storage. Marking of slow cache lines can be done, as in [1] and [3], by clearing their *valid* bit and setting their *lock* bit. Finally, at runtime the cache works as ever without noticing and without using the slow lines.

3 Problem Definition and Algorithm

The following notations are added to those in Section 2.3:

***w*:** the original number of ways in the cache.

***b*:** the number of bits per cache line (including tag bits).

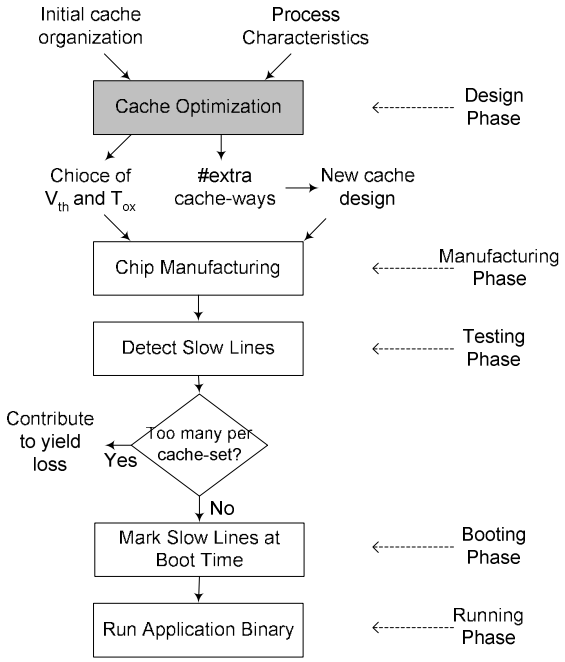

Figure 5. Big picture of our proposed approach.

s: the number of cache-sets.
***P_L*:** the leakage power of the cache.
***V_{th}*:** the optimal value for the V_{th} of cache transistors.
***T_{ox}*:** the optimal gate-oxide thickness of cache transistors.
The optimization problem can be formally defined as follows:

"For a given process technology (i.e. μ_d, σ_d), cache organization (i.e. w, b, and s), and timing-yield (Y), minimize the leakage power of the cache (i.e. P_L) by setting V_{th}, T_{ox}, and N such that the target delay, D , is kept unchanged."

Algorithm. The following algorithm takes the cache organization and process technology as input and provides the best choice of V_{th}, T_{ox}, and N if our technique can be useful. Otherwise, it returns an empty set indicating the failure.

```
Algorithm 1: OptimizeCacheDesign()
Inputs: (σd, μd: process technology characteristics),
        (w, b, s: original cache configuration),
        (Y: Target timing-yield of cache)
Output: set of (N, Vth, Tox) triples.
1 set answers_set = empty_set
2 compute D based on Y, σd, μd.
3 compute PL (leakage power) of the original cache.
4 for N=1 to w/2 do
4.1 compute the highest μd in presence of N extra
    cache ways such that D and Y are kept intact.
4.2 choose the best Vth and Tox corresponding to this
    new μd
4.3 compute PL' (leakage power) of the new cache
4.4 if PL'<PL then add (N, Vth, Tox) to answers_set
5 endfor
6 return answers_set
```

The algorithm simply assumes various values for N (line 4), and then checks whether the corresponding V_{th} and T_{ox} (lines 4.1 and 4.2) sufficiently reduce the leakage so that the static power of the cache with the extra way(s) is less than before or not (lines 4.3 and 4.4). To compute the new μ_d in line 4.1, first yield formulas (similar to (2) and (3) in Section 2.3 depending on the cache organization parameters: w, b, and s) are used to determine the Y_{cell} corresponding to N. Then the corresponding μ_d for the given D and just-calculated Y_{cell} is computed using (1). In order to choose the best values of V_{th} and T_{ox} at step 4.2, leakage vs. delay curves are used (see Figure 4); these curves are obtained through SPICE simulations of the entire cache design for the new σ_d and various values of V_{th} and T_{ox}. Given the new mean delay (μ_d), the curve giving the least leakage is chosen and its corresponding V_{th} and T_{ox} reflect the best choice. The static power of the cache without and with extra cache-ways (lines 3 and 4.3 of the algorithm) are also computed from the above-mentioned leakage-delay curves.

4 Experimental Results

We applied our technique to a real 90nm middle-performance process technology (undisclosed due to NDA) with 320mV nominal V_{th} and 1V supply voltage. We developed a cache memory design and obtained its leakage power using SPICE Monte Carlo simulation to take variation into account. Results of running our algorithm on a 16KB cache (4-way set-associative, 256 sets, 128-bit line-size, and 20 bits per tag) are given in Table 1. The table shows that our technique can save more when targeting a higher yield as well as when there is a higher delay variation in the manufacturing process. The algorithm execution time is just a fraction of a second on a Xeon 3.80GHz processor with 2MB of cache and 3.5GB of memory, but it took us a week to run all SPICE simulations on the same machine so as to obtain the tables used in step 4.2 of the algorithm (see Figure 4); however, noting that this is done only once for entire manufacturing of all the cache chips, this execution-time would not be a limiting factor.

Table 1. Leakage reduction results using our technique

Target Yield	Delay variation	Cache leakage power (μW)			Saving (%)	
		original	N=1	N=2	N=1	N=2
90%	3%	20.552	18.030	18.773	12.27	8.66
	5%	149.80	89.123	81.191	40.51	45.80
95%	3%	21.592	18.619	19.209	13.77	11.04
	5%	166.89	95.153	84.500	42.98	49.37
99%	3%	24.501	20.014	20.171	18.32	17.67
	5%	203.29	111.83	93.502	44.99	54.00

Our technique focuses on leakage power reduction, but imposes some overhead on the dynamic energy of cache access due to the additional cache ways. Thus, the total power reduction depends on the actual share of leakage in total power consumption. We conducted experiments to measure the ratio of static to dynamic power in an embedded processors, M32R-II, when running a number of benchmarks. Our M32R-II system has separate level-1 instruction and data caches: a 16KB 4-way set-associative cache with 32-byte lines for instructions and the same organization but 16-byte cache-lines for data. It is also equipped with a 16KB unified level-2 cache with the same organization as the level-1 instruction-cache. We used four applications from MiBench benchmark suit and compiled them once with no compiler option and once with '-O3' full-optimization option. Table 2 gives the number of static instructions in each benchmark along with the number of misses, the number of all-cache-ways-activated accesses (far less than total executed instructions; achieved by implementing ILWM technique [14]), and the number of M32R-II clock cycles for executing the instructions (obtained from RTL simulation). We simulated 1 million instructions of each benchmark to gather cache access statistics. Dynamic energy per cache-access as well as cache-access delay are separately obtained from SPICE simulation of the entire cache using the above-mentioned 90nm transistor model.

Table 2. Characteristics of the benchmarks.

Benchmark	Code size (#instructions)	#I-misses	#full-way accesses	#clocks
JPEG Enc.	151916	830	154234	2322585
JPEG(-O3)	147563	616	129701	1655925
MPEG2 Enc.	175190	499	181135	1551097
MPEG2 (-O3)	167629	475	186507	1540983
FFT	118639	239	182304	1542659
FFT (-O3)	118517	231	166729	1529172
Compress	195650	66	144368	2427196
Compress (-O3)	195391	40	170621	2145614

Table 3 gives the original dynamic energy per access of M32R instruction cache as well as in presence of one and two extra cache ways; note that when only one cache way is accessed using ILWM, dynamic energy per access is the same for all caches. The table also gives leakage power of the caches in presence of 5% delay variation; values for extra

cache ways (N=1, 2) are the leakage power after applying our technique targeting 99% timing-yield.

Table 3. Dynamic energy of each cache access and static power of cache before and after applying our technique.

	16KB, 4-way cache (128 sets)		
	original	N=1	N=2
Energy per single-way access (pJ)	4.04	4.04	4.04
Energy per full-ways access (pJ)	16.79	20.75	24.71
Leakage at 5% variation (μW)	406.58	223.65	187.00

Figure 6 shows the breakdown of the L1 instruction-cache energy consumption when running our benchmarks on M32R-II processor assuming 5% delay variation and 99% target timing-yield. The results reported here include ILWM technique [14] (see Section 2.1). Several other techniques such as [16] and those in [5] can also be used to further reduce dynamic power; these are orthogonal to our technique and can be combined with it to further reduce total energy.

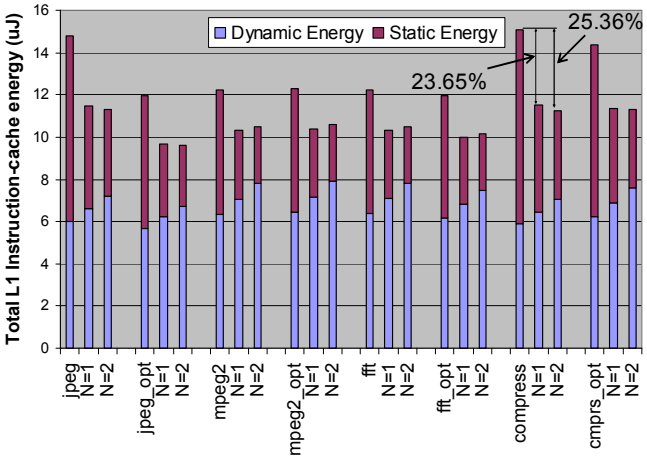

Figure 6. Results on L1 instruction-cache.

Leakage energy comprises a bigger part in L2 caches since such caches are less frequently accessed and in general are larger than L1 caches. Results of applying our technique to L2 cache for the same processor and benchmarks are given in Figure 7, which confirms the above expectation. The cache is the same 16KB one, but is unified for instructions and data.

5 Summary and Conclusion

We presented a design- and manufacturing-time optimization technique to reduce leakage power dissipation, without sacrificing performance or timing-yield nor capacity, of cache memories in nanometer technologies where leakage is dominating the total power and where high process variation is observed; our technique reduces leakage by choosing higher V_{th} and T_{ox}, and then adds extra cache ways to compensate for the cache-lines that would now violate the cache target delay. Our technique achieved up to 53.37% reduction in total cache power on a commercial 90nm technology. Although we substantially increase the chip area by adding extra cache ways, it is not a problem given the density capability of today processes especially since we reduce total power despite the additional transistors and area. Noting that leakage power rises in general in finer technologies, and moreover, it exponentially

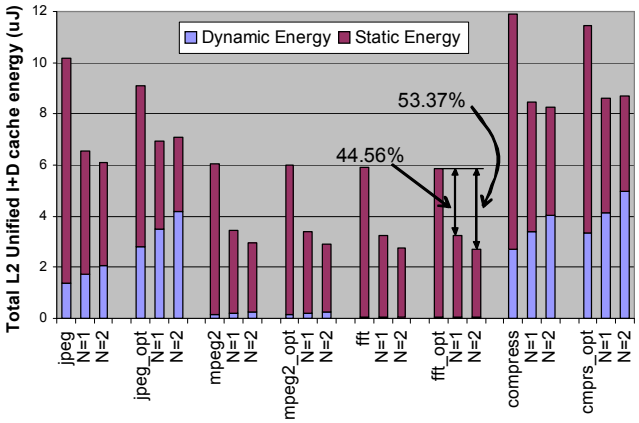

Figure 7. Experimental results on L2 unified cache.

increases with temperature while dynamic power decreases with technology scaling, this technique will be even more effective in future technologies.

Acknowledgments

This work is supported by VLSI Design and Education Center (VDEC), The University of Tokyo with the collaboration of STARC, Panasonic, NEC Electronics, Renesas Technology, and Toshiba. This work is also supported by Core Research for Evolutional Science and Technology (CREST) project of Japan Science and Technology Corporation (JST). We are grateful for their support.

References

[1] A. Agarwal, B.C. Paul, H. Mahmoodi, A. Datta and K. Roy, "A process-tolerant cache architecture for improved yield in nanoscale technologies," *IEEE Trans. on VLSI,* vol. 13, no. 1, pp. 27-38, 2005.

[2] P.P. Shirvani and E.J. McCluskey, "PADded cache: a new fault-tolerance technique for cache memories," *Proc. IEEE VLSI Test Symp.*, pp. 440-445, April 1999.

[3] T. Ishihara, F. Fallah, "A cache-defect-aware code placement algorithm for improving the performance of processors," *Proc. Int'l Conference on Computer-Aided Design*, pp. 995-1001, 2005.

[4] R. Panwar and D. Rennels, "Reducing the frequency of tag compares for low-power i_cache design," *Proc. Int. Symp. on Low-Power Electronics and Design*, pp. 57-62, 1995.

[5] V.G. Moshnyaga, K. Inoue, "Low-Power Cache Design," in Low-Power Electronics Design, C. Piguet Eds., CRC Press, 2005.

[6] S. Kaxiras, Z. Hu, M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," *Int'l Symposium on Computer Architecture,* pp. 240-251, 2001.

[7] K. Flautner, et al., "Drowsy caches: simple techniques for reducing leakage power," *Proc. Int'l Symp. on Computer Architecture,* 2002.

[8] N.H.E. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, Addison Wesley, 2004.

[9] M.D. Powell, et al., "Gated-V_{dd}: a circuit technique to reduce leakage in cache memories," *Int'l Symp. Low Power Elec. and Design*, 2000.

[10] F. Fallah, M. Pedram, "Circuit and system level power management," in *Power Aware Design Methodologies*, M. Pedram and J. Rabaey Eds., Kluwer Academic Pub., pp. 373-412, 2002.

[11] S. Narendra, et. al, "1.1 V 1 GHz communications router with on-chip body bias in 150nm CMOS," *Proc. IEEE ISSCC Dig. Tech. Papers,* Feb. 2002, pp. 270-271.

[12] T. Kuroda, T. Fujita, F. Hatori, and T. Sakurai, "Variable threshold-voltage CMOS technology," *IEICE Trans. on Fund. of Electronics, Communications and Computer Sciences,* vol. E83-C, 2000.

[13] K. Kanda, N. Duc Minh, H. Kawaguchi, and T. Sakurai, "Abnormal leakage suppression (ALS) scheme for low standby current SRAMs," *Proc. IEEE International Solid-State Circuits Conference*, 2001,.

[14] R. Panwar and D. Rennels, "Reducing the frequency of tag compares for low-power I-cache design," *Proc. International Symposium Low Power Electronics and Design*, pp. 57-62, 1995.

[15] M. Mullar, "Power efficiency and low cost: the ARM6 family, *Proc. Hot Chips IV,* 1992.

[16] T. Ishihara, F. Fallah, "A non-uniform cache architecture for low power system design," *Proc. International Symposium Low Power Electronics and Design*, 2005.