Process Variation Aware Cache Leakage Management

Ke Meng
k-meng@northwestern.edu

Russ Joseph
rjoseph@eecs.northwestern.edu

Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

ABSTRACT

In a few technology generations, limitations of fabrication processes will make accurate design time power estimates a daunting challenge. Static leakage current which comprises a significant fraction of total power due to large on-chip caches, is exponentially dependent on widely varying physical parameters such as gate length, gate oxide thickness, and dopant ion concentration. In large structures like on-chip caches, this may mean that one portion of a cache may consume an order of magnitude larger static power than equivalently sized regions.

Under this climate, egalitarian management of physical resources is clearly untenable. In this paper, we analyze the effects of within-die and die-to-die leakage variation for on-chip caches. We then propose *way prioritization*, a manufacturing variation aware scheme that minimizes cache leakage energy. Our results show that significant average power reductions are possible without undue hardware complexity or performance compromise.

Categories and Subject Descriptors

B.3.2 [**Memory Structures**]: Design Styles - *cache memories*

General Terms

Design, Performance, Experimentation

Keywords

process variation, low power, leakage, cache management, Gated-VDD, selective cache ways

1. INTRODUCTION

In future technology generations, the promise of burgeoning transistor budgets will be tempered by a sobering fact: foundries will have increasingly limited control of transistor quality. ITRS lists several fundamental challenges for chip manufacture which limit fabrication accuracy but have no known solutions [15]. These quality control issues manifest themselves as randomly distributed variations in characteristics such as transistor length and gate oxide thickness [4]. Ultimately, these variations affect power and performance by altering circuit behavior, leading to physical implementations which may no longer meet design specifications

Leakage current, well known as a prominent threat for deep submicron design, is highly susceptible to manufacturing variations and will consequently be more difficult to optimize [7]. Recent industry data show as much as a 20× variation in total chip leakage power for dies from the same wafer [4]. The majority of this variation comes from deviations in leakage power. As a result of the exponential dependence on gate length (L) and threshold voltage (v_t), which vary on a die-to-die and within-die basis, even schematically equivalent cells on the same chip can have dramatically different leakage power.

Due to the huge potential variances in device characteristics, microarchitectures can no longer be designed with an egalitarian view of hardware resources. In particular, fixed parameter leakage models and optimizations will become increasingly less effective because they can only target the fraction of chips which have no imperfections, ignoring the majority of chips which have significant on-chip variations. The consequences will be significant for cache memories due to their large on-chip area. First, they will account for a large portion of the total chip leakage. Second, they will be exposed to significant amounts of spatially dependent parameter variation. Although techniques like adaptive body biasing ([16]) have potential for reducing leakage, they necessitate use of complicated circuit techniques which may not be feasible for fine grain spatial leakage control.

In this work, we examine the effects that leakage variation can have on cache memories. This paper makes two primary contributions:

- We develop architecture-level statistical models and methodology for studying cache leakage under manufacturing variations. We are among the first to survey the local and global effects of manufacturing variation on cache leakage.
- Based on our analysis, we introduce *way prioritization*, a leakage reduction strategy that appropriately sizes caches to reduce the average and worst case leakage power without compromising performance. Overall, we show a 28% reduction in L3 leakage power consumption versus a variation oblivious resizing strategy.

The remainder of this paper is organized as follows: Section 2 introduces the statistical models used in this paper. In Section 3, we propose a cache organization which accounts for process variation to improve leakage management. Experimental methodology and evaluation of way prioritization follow in Sections 4 and 5. Finally, we conclude with a summary in Section 6.

ISLPED'06, October 4–6, 2006, Tegernsee, Germany.

2. MODELING PROCESS VARIATION

In this paper, we leverage existing stochastic techniques common in the circuit design community to develop architectural models for on-chip caches. Our approach, depicted in Figure 1, uses two phases. In the first, we construct a statistical model for a cache based on organizational parameters such as capacity and block size, as well as physical parameters such as geometric position on chip and overall die size. For a fixed set of manufacturing quality parameters, this model captures the local leakage variation within different regions of the cache and the total cache leakage variation across dies. In the second phase, we conduct Monte Carlo analysis by generating multiple random samples which have the statistical properties described by our model. While Monte Carlo simulation is not as elegant as some recently proposed analytic strategies for computing total leakage [6, 14], our methodology experiments allow us to examine several spatial aspects of the problem that the analytic techniques do not support.

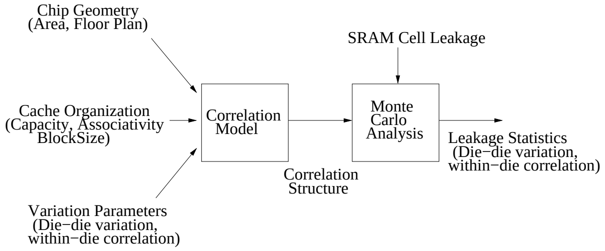


Figure 1: Leakage variation modeling approach used in this work.

2.1 Leakage Model for An Individual SRAM Cell

A six-transistor (6-T) SRAM cell serves as the basic construction block of a standard cache design. We develop a 6-T leakage model as the basis of our static power analysis. In current CMOS technologies, leakage current is composed of three major sources: subthreshold leakage, gate leakage and substrate leakage. SPICE simulation with PTM [5] 32nm technology predictive model card identifies subthreshold leakage as the dominant source for near future technologies. In the rest of this paper, we focus on subthreshold leakage, but the methodology is readily extendable to other types of leakage.

The basis of a 6-T cell is a pair of cross-coupled inverters. Subthreshold leakage occurs on a PMOS transistor and an NMOS transistor from each inverter regardless of the value stored in the cell. As shown by Rao et al., gate length variation plays a dominant role in subthreshold leakage [14]. In this work, we focus on the leakage current deviations caused by gate length variation.

Rao et al. took the following empirical equation to model the subthreshold leakage current of a single transistor with respect to the transistor gate length [14]:

$$I = p_1 e^{(p_2 L + p_3 L^2)} \tag{1}$$

We extend the usage of the above equation to a SRAM cell. First we create a 6-T SPICE model, then we collect its leakage current during a series of simulations where we sweep transistor gate lengths over the $+/-10\%$ range of their nominal values. We then apply curve fitting to identify the constants. In our simulations, we assume a perfect correlation between gate lengths in an individual SRAM cell due to their proximity. These are common assumptions found in the literature [1, 14].

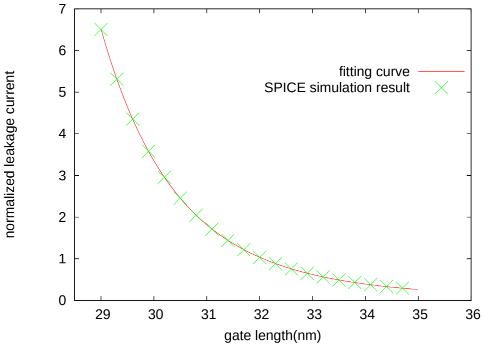


Figure 2: SPICE simulation and Curve fitting results.

Figure 2 shows the comparison between the leakage value obtained from SPICE simulation and the fitting curve. The fit matches the simulation results accurately over a wide range.

The fabricated gate length is the sum of nominal value and the variation:

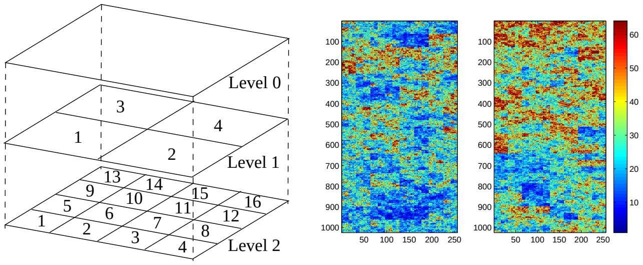
$$L_{total} = L_{nominal} + L_{intradie\_variation} + L_{interdie\_variation} \tag{2}$$

Intra-die variation expresses the difference in gate length from a transistor to another transistor on a single die, while inter-die variation refers to the die-to-die variation. We model both variations as normally distributed random variables $L_{interdie_variation}$and $L_{intradie\_variation}$. In the following section, we discuss the gate length correlation patterns which form the basis for the variation models.

2.2 Spatial Correlation Between SRAM Cells

Experimental results from physical measurements in [10] show that strong spatial correlation exists on intra-die gate length variations. For two identical transistors, the correlation factor of their gate length drops almost linearly as the distance between the two devices increases. The correlation nears zero as the separation approaches half the width of the die. This spatial correlation is also independent of chip size if one chip is printed per field. The consequence is that there is physical-spatial locality for leakage in regular array structures such as caches.

To model the spatial correlation in a die, we use the hierarchical correlation modeling method introduced in [1] to capture within-die variation. Figure 3(a) illustrates how the hierarchy corresponds to physical spacing for a small example. At the top level, we generate a random variable which is an additive term present in the gate length of all devices. We recursively divide the area into quadrants and generate random additive terms for each quadrant. The effective gate length for specific device can be found by adding the random variables in its hierarchy to the nominal gate length. In our experiments, we choose a grid with an area of 32 6-T cells as



(a) Hierarchal correlation matrix (b) Correlated leakage variation

Figure 3: Construction of correlation models used in this work and resulting leakage variation for two samples.

the terminal size and end the recursion there. Consequently, for a 25% chip area 16MB data array, we have a total of 12 levels and 4,194,304 atomic macro cells over the cache data array. By scaling the additive terms as the grid size decreases, we can match the spatial correlation presented in [10].

Figure 3(b) shows an example of how severe the leakage variation can be in cache structures. The same batch of simulation runs also featured a maximum 20× cache leakage variation from die-to-die. The implication is that it may no longer be wise to treat two chips, two processor cores on the same chip, or even two sub-arrays in the same cache identically. It is important to recognize the physical differences rooted in process variation. Additional power savings can be gained if we can find ways to manage a processor on a granularity fine enough to isolate spatial variations.

3. PROCESS VARIATION AWARE CACHE SIZING

Because caches normally occupy a large portion of on-chip area, they are a major target for static power control. The regularity of caches provide a large design space and flexibility for cache structure design. In this section, we propose a cache design with leakage variation awareness.

3.1 Enabling Techniques

Previous work on power aware processor design has examined several methods for reducing both dynamic and static power for caches [2, 11, 13, 9]. Our approach extends two widely known techniques, which we briefly summarize:

- **Selective Cache Ways [2]** - In an n-way set associative cache, one or more ways (regular sub-arrays) can be disabled, effectively reducing both the associativity and the overall size of the cache. Power saving modes can then be applied to decoders, precharge structures, sense amplifiers, and SRAM cells in the disabled ways.

- **Gated VDD [13]** - Portions of logic or memory can be selectively enabled or disabled by placing a controlling transistor on conductive paths to VDD/VSS. Leakage current can be effectively eliminated in SRAM cells via this technique.

We use cache ways as the granularity at which to apply VDD gating. With these two technologies, caches can be effectively sized to meet workload demands.

3.2 Way Prioritization

Conventional cache sizing strategies do not differentiate cache ways, despite the fact that some portions of the cache will be leakier than others. This egalitarian view of the cache may not achieve good energy savings if one of the enabled ways happens to have a high overall leakage current. Ideally, we wish to only use cache ways that have energy costs which are commensurate with the performance benefit from having increased cache size.

We propose *way prioritization* as a technique to enable the appropriate number of cache ways and select which subset of the cache array to make active. Way prioritization achieves good overall energy savings for a given performance level by accounting for the effects of within-die and die-to-die variation. Just as in the standard selective ways approach, the cache needs to be appropriately sized for a given workload. Furthermore, way prioritization can be applied in concert with circuit-level leakage and variation aware design techniques such as adaptive body-biasing [16] and multiple v_t assignment to maximize leakage savings. In this section, we focus on the mechanisms that allow us to isolate and prioritize ways. We discuss tradeoffs in sizing policies in the subsequent section. Under way prioritization the SRAM cells belonging to the same way are organized into a sub-array on the chip. In our case, this has the additional benefit of taking advantage of the spatial leakage correlation.

The key hardware difference between a standard selective cache ways implementation and a way prioritized one is a set of hardware registers which identify the leaky cache ways and make cache sizing effective. Figure 4 depicts the hardware differentiation and highlights the PRIORITY and DEGREE registers. The PRIORITY register consists of n entries of $log_2 n$ bit size for an n-way set associative cache. The entries in the PRIORITY register correspond to physical ways sorted by descending leakage power. The DEGREE register supplements this information by tracking the absolute leakage of the corresponding physical way. When the cache is being resized for a particular workload, these registers can be queried to determine *how many* ways should be enabled and *which specific* ways should be enabled.

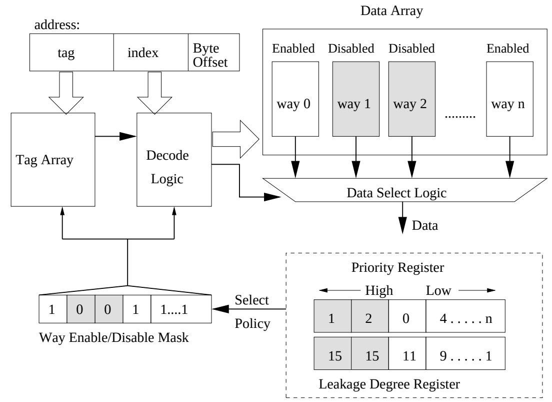


Figure 4: Hardware organization for prioritized cache ways

The measurements needed to populate the leakage registers can be collected off-line during the manufacturing test phase. Individual cache ways can be independently enabled as part of a built in self-test (BIST) sequence while the rest of the processor is left idle. The leakage current for each way can be calculated from ammeter readings of total chip current draw. Collected data can be quantized, physical ways can be sorted by their leakage power, and the resultant information can be kept in non-volatile near-chip storage. At boot time, the PRIORITY and DEGREE registers can be configured based on the previously determined values.

3.3 Cache Sizing Policies for Individual Chips and Workloads

Way prioritization allows the cache to be sized and configured on a per workload and per chip basis. Given knowledge of how application performance varies with increasing total cache size, we can either chose a sizing which minimizes power for a fixed performance level or we can target a more flexible power/performance optimization metric (e.g. energy-delay product). We assume that static profiling [2] or dynamic working set analysis [8] can identify how performance scales with cache size. The remainder of this discussion focuses on policies.

For a fixed performance level, for example 2% slowdown, a working set profile provides an appropriate cache size, or equivalently, k, the optimal number of active ways. Under way prioritization, the cache can be resized by enabling the last k physical ways held in the PRIORITY register and disabling all other ways. Any dirty data held in a newly disabled way is written back to the next level in the memory hierarchy before VDD is finally gated. Because the physical ways are pre-sorted by leakage, this cache configuration represents the organization which would produce the minimal leakage power for that given performance level.

For optimizations which allow a variable amount of performance degradation, we need to know the incremental energy cost of enabling each additional way. Different physical chips may have different total leakage or different ratios of leakage between ways. The DEGREE register tracks how much leakage energy each additional way contributes. When reconfiguring the cache to minimize energy-delay, for example, the optimal value can be found by iterating through the PRIORITY and DEGREE registers. Initially, the core power would be added to the leakage power for the least leaky way and that sum would be multiplied by $slowdown^2$ for the minimal cache size. For each additional way, the total chip power increases, but the delay (obtained via profiling) decreases. The minimal energy-delay product can be found by repeating this process. If core power increases and delay decreases monotonically with cache size, the search process need not be exhaustive: an increase in energy-delay signals that the optimal sizing has already been reached. In the worst case, none of the ways are disabled, and a total of n iterations are performed. Note that the pre-sorting of the PRIORITY register means that we do not need to identify all possible combinations of physical ways. Because this sizing process is slightly more complicated than trying to meet a fixed performance goal, it may be preferable to implement it in microcode or the operating system.

In addition to manufacturing variations in gate length, runtime parameters like temperature can also have an influence on leakage power. Our work focuses primarily on large L3 caches which typically have lower peak temperatures, so we believe that the thermal contribution to variation will be relatively small. However, on-chip thermal sensors could be combined with structures like the DEGREE register to adjust power estimates for runtime conditions. We plan to investigate this approach in future work.

4. EXPERIMENTAL METHODOLOGY

4.1 Processor Model

Our experiments model a high-performance server class processor comparable to the Intel Montecito [12]. The processor features two symmetric cores, each of which has private L1, L2, and L3 caches. Table 1 summarizes the pipeline configuration and cache organization for our processor. The processor executes the Alpha ISA.

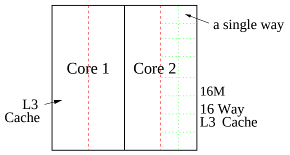


Figure 5: The base floorplan for performance simulation

We target this design for a future 32nm technology process where leakage and process variation will play prominent roles. We devise the floorplan for this design by assuming that the footprint for a single processor core is equivalent to that of a single core for Montecito in a 90nm process. We then scale the core to 32nm and assume a base floorplan for our experiments as presented in Figure 5. The cache structure studied is a re-sizable 16M L3 Cache which occupies 25% of the total chip area.

The processor is modeled via the M5 Full System simulator [3] which includes detailed models of pipelines, caches, buses, and off-chip memory. The simulator runs a slightly modified version of the linux-2.6.8.1 kernel and captures all of the performance effects of the multiprogrammed workloads used in this study.

		Workload	
Single Core		Name	Apps
Clock	2.5 GHz	int.2.1	bzip2, crafty
Fetch/Decode	4 inst	int.2.2	eon, twolf
Issue	6 inst, out-of-order	fp.2.1	ammp, art
IQ/LSQ/ROB	32/40/80 entries	fp.2.2	equake, mesa
Func Units	4 IntALU, 1 IntMul	mix2.1	ammp, bzip2
	1 FPALU, 1 FPMul	mix2.2	art, crafty
	2 MemPorts	mix2.3	equake, eon
L1 I-Cache	16KB 4-w 1 cyc	mix2.4	mesa, twolf
L1 D-Cache	16KB 4-w 2 cyc		
L2 I-Cache	1MB 8-w 7 cyc		
L2 D-Cache	256KB 4-w 6 cyc		
L3 Cache	16MB 16-w 20 cyc		

Table 1: Processor parameters for a single core and workloads used in this study.

4.2 Workloads

To evaluate the efficacy of our leakage management approach, we use several multiprogrammed workloads which showcase a variety of memory usage patterns. Individual applications are taken from the SPEC CPU2000 benchmark suite. To reduce the total number of simulations, we identify a subset of SPEC applications which exhibit a range of performance characteristics and then group them into eight mix sets. Table 1 also shows the workload groupings used in the experiments. The benchmarks were compiled with `gcc-4.0.1` for `alpha-linux` using `-O4 -ffast-math -funroll-loops` flags. All of our workloads run on a single core of our dual core design. We report performance and power savings with respect to a single core.

5. RESULTS

5.1 Leakage Variation in Cache Structures

We used our Monte Carlo method to evaluate spatial leakage variation under several different cache configurations. Each simulation point consists of 10,000 samples. In all cases we assumed normal distribution on gate length intra-die variation and the 3σ value of the distribution was set to 9.4% of the nominal value.

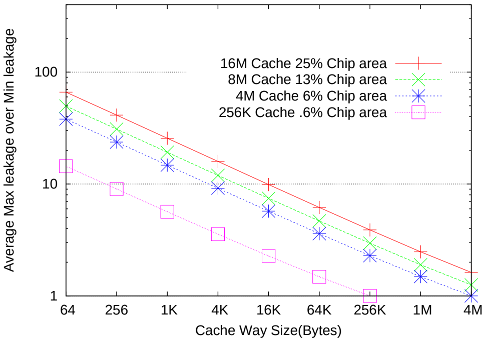


Figure 6: Average max leakage over min leakage with respect to way sizes.

Figure 6 shows the dramatic difference in cache leakage for regions chosen from different locations in the data array. This log-log plot tracks the ratio of maximum to minimum leakage power for cache regions of varying size. For each region size, we first select square sub-arrays of the cache which have maximum and minimum leakage, and then we compute their ratio. We draw these curves for a wide range of cache sizes. From Figure 6, we first see that the leakage ratio decreases rapidly as the region size increases. This is due to the fact that when the regions are small, there are many distant sections to choose from, increasing the chance that the regions do not have similar parameter sizes. As the regions grow, both the maximum and minimum leakage regions tend towards mean values, and the distance between the regions decreases. The second trend is that increasing cache sizes boost the max/min ratio. This is due to the fact that larger caches have a larger population of 6-T cells and hence longer "tails" on the distribution. We can extrapolate the benefits of way prioritization if we consider cache ways to be our regions. First, the savings are likely maximized on very large caches. Second, greater associativity and hence a larger number of small cache ways increase the potential for power savings. In the remainder of this paper, we focus our analysis on a large, highly associative cache (16-way 16M L3).

Again employing our Monte Carlo approach, we create 10,000 random samples of the L3 cache described in Section 4, assuming zero global inter-die variation. For each run, we sorted the cache ways by decreasing leakage current. Figure 7 shows the normalized mean leakage current for the ranked cache ways and the standard deviation spread. We can see that on average the leakiest way consumes 2.44 times more static power than the least leaky way. Figure 7 also shows way leakage current for several samples. Clearly, different chips can have divergent leakage profiles for their ways.

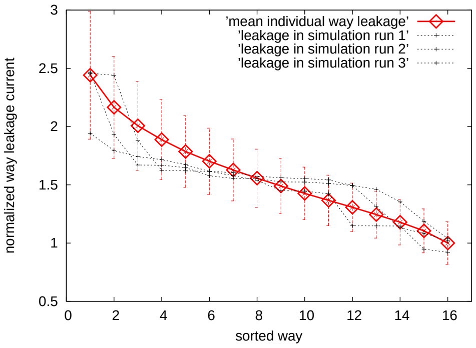


Figure 7: Normalized mean leakage current from most leaky to least leaky ways.

Inter-die Variation 3σ Perc. of Nominal Value		Normalized Way Leakage Rank(Decreasing Leakage)			
		1	6	10	16
0%	mean	2.44	1.70	1.43	1.00
	stdev	0.55	0.29	0.23	0.18
2.8%	mean	2.59	1.80	1.50	1.05
	stdev	1.10	0.67	0.54	0.37
5.6%	mean	2.75	1.90	1.58	1.10
	stdev	1.57	0.98	0.79	0.52
9.4%	mean	3.05	2.09	1.73	1.19
	stdev	2.40	1.50	1.20	0.78

Table 2: The influence of inter-die variation on way leakage current distribution.

Table 2 lists the normalized mean leakage currents and standard deviation of the leakage currents calculated from the simulation results of leakage-severity-sorted ways. This time the 3σ value of inter-die variation changes from 0% to 9.4% and both the mean leakage and standard deviation show an increase as the inter-die variation increases. The increase for mean leakage is quite limited while the standard deviation sees a significant jump. It indicates that we can expect much more variance in way leakage profiles and a better savings from way prioritization.

5.2 Energy Savings of Variation Aware Sizing

Figure 8 shows the benefits of way prioritization as applied to the L3 cache for multiprogrammed workloads. In these experiments, we again assume zero inter-die variation and a 3σ 9.4% intra-die variation. All the benchmarks place small to moderate pressure on the L3 Cache. We assume a static cache sizing policy which chooses the maximum number of ways to disable while not reducing performance by more than 2%.

In Figure 8, the y-axis represents the static power consumed by working ways as a percentage of the total static power of the L3 Cache. Clearly, our method provides satisfying power saving over standard selective ways (for both the average and worst case). Prioritized ways consume an average 28% and 48% less cache static power over the non-variation aware mean and worst-case scenarios.

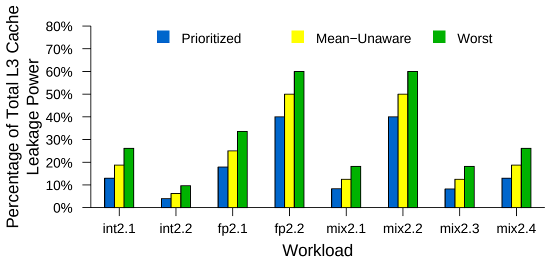


Figure 8: Cache leakage energy of way prioritization (prioritized) versus variation unaware selective cache ways in average (mean-unaware) and pathologically bad (worst) cases.

Results based on the energy-delay metric are presented in Figure 9. We assume that the cache static leakage power comprises 20% of the total core power consumption and that the dynamic power of the whole core is unchanged when cache ways are closed. We normalize the results to the full cache leakage power. In theory, the best energy-delay product achievable is 0.8, when all ways are closed and the performance does not suffer. The results in Figure 9 indicate we can achieve better energy-delay product than mean-unaware case in which all ways are assumed to consume the same amount of power and the worst-case scenario in which the most leaky ways are chosen to stay on.

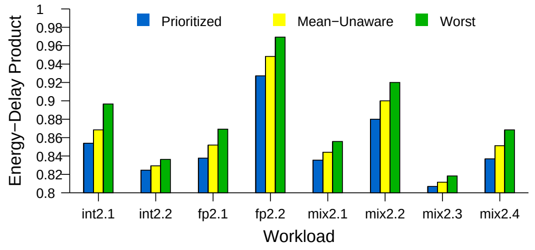


Figure 9: Energy-Delay product of way prioritization (prioritized) versus variation unaware average (mean-unaware) and pathologically bad (worst) cases.

A relevant feature not captured in the above figures is that a single mean leakage current profile cannot represent actual leakage profiles. In certain situations a variation oblivious cache-sizing method could make a faulty decision in closing ways. In addition to picking the wrong physical ways to close, unaware sizing risks choosing the wrong number of ways to close. By taking into account the individual leakage profile of a chip, way prioritization always chooses the best energy-delay sizing.

6. CONCLUSION

In future technologies, leakage power will comprise a significant portion of the total chip power and the effects of manufacturing variations will be pronounced. This paper is among the first to examine the effects that parameter variation have on cache leakage. Our findings show that spatial variation can have significant influence on the cache leakage profile. In particular, equal areas of the cache may have leakage factors that differ by more than an order of magnitude. We propose way prioritization, a cache organization that considers the leakage profile of a chip and resizes the cache by closing sub-arrays that have higher leakage factors. The result is a substantial energy savings with little impact on complexity or performance. Furthermore, because way prioritization works at the microarchitectural level, it is complementary to existing circuit-level leakage reduction and variation tolerance schemes.

7. REFERENCES

[1] A. Agarwal, D. Blaauw, S. Sundareswaran, V. Zolotov, M. Zhou, K. Gala, and R. Panda. Path-based statistical timing analysis considering inter and intra-die correlations. In *Proc. of TAU*, 2002.

[2] D. H. Albonesi. Selective cache ways: On-demand cache resource allocation. In *Proc. of the 32nd Annual IEEE/ACM Int. Symp. on Microarchitecture*, pages 248–259, Nov. 1999.

[3] N. L. Binkert, E. G. Hallnor, and S. K. Reinhardt. Network-oriented full-system simulation using m5. In *6th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Feb. 2003.

[4] S. Borkar et al. Parameter variations and impact on circuits and microarchitecture. In *Proc. of the 40th DAC*, 2003.

[5] Y. Cao, D. S. T. Sato, M. Orshansky, and C. Hu. New paradigm of predictive mosfet and interconnect modeling for early circuit design. In *Proc. of CICC*, pages 201–204, 2000. http://www.eas.asu.edu/p̃tm.

[6] H. Chang and S. Sapatnekar. Full-chip analysis of leakage power under process variations - including spatial correlations. In *Proceedings of the ACM/IEEE Design Automation Conference*, 2005.

[7] A. Devgan and S. Nassif. Power variability and its impact on design. In *Proceedings of the 18th International Conference on VLSI Design (VLSID-05)*, 2005.

[8] A. Dhodapkar and J. E. Smith. Managing multi-configuration hardware via dynamic working set analysis. In *Proc. of 29th Int. Symp. on Computer Architecture (ISCA-29)*, May 2002.

[9] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: Simple techniques for reducing leakage power. In *Proceedings of 29th International Symposium on Computer Architecture (ISCA-29)*, May 2002.

[10] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos. Modeling within-die spatial correlation effects for process-design co-optimization. In *Proc. of the 6th Int. Symp. on Quality Electronic Design*, 2005.

[11] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th International Symposium on Computer Architecture (ISCA-28)*, June 2001.

[12] C. McNairy and R. Bhatia. Montecito: A dual-core dual-thread itanium processor. *IEEE Micro*, 25:10–20, Apr. 2005.

[13] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. Vijaykumar. Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *ACM/IEEE ISLPED*, 2000.

[14] D. B. Rajeev Rao, Ashish Srivastava and D. Sylvester. Statistical analysis of subthreshold leakage current for vlsi circuits. *IEEE Trans. on VLSI Systems*, 12:131–139, Feb. 2004.

[15] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, 2003. http://public.itrs.net/Files/2003ITRS/Home.htm.

[16] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, Fellow, A. P. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 37:1396–1402, Nov. 2002.