Presented at the 2006 P=ac2 Conference, IBM T.J. Watson Research Center, Yorktown, NY.

Physical Resource Matching Under Power Asymmetry

Ke Meng Frank Huebbers Russ Joseph Yehea Ismail

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

Abstract

Manufacturing related variations are predicted to have a significant impact on power consumption for near future technology generations. Due to both die-to-die and within-die parameter variations, the power consumption of identically designed hardware resources can vary randomly on a per instance basis. This introduces fabrication induced asymmetry, which can undermine basic assumptions about power/performance balances in a design. Specifically, architects expect that superscalar pipeline resources and cores in a homogenous CMP design are truly homogenous when fabricated on a chip. In this paper, we describe implications that fabrication power asymmetry have on a CMP design with respect to the allocation and utilization of hardware resources and the mapping between applications and physical cores. We introduce hardware/software management techniques that can overcome power asymmetry by considering the dissimilar power profiles of physical resources under parameter variation.

1 Introduction

For the foreseeable future, manufacturing variations will have severe consequences on the performance and power consumption of high-performance microprocessor designs [4, 8]. The fabrication process introduces random and systematic variations in physical device characteristics including gate length, gate width, oxide thickness, and dopant ion concentration [26]. These in turn impact electrical characteristics of transistors, most prominently, v_t. This has drastic effects on performance and power. These factors are already present in commercially available chips [4], and technology forecasts paint a grim future [29]. Process induced variation creates as much as a 30% deviation in operating frequency and a staggering $20\times$ leakage current variation in less-than-cutting-edge 180nm technology [4]. This is particularly troubling because power density concerns that raise cooling requirements and acerbate in-field failures have already become first class design constraints for high-performance design [5, 24, 31, 32, 33] and are expected to worsen [29]. Due to the limited controllability of physical parameter deviations at chip manufacture, it will be increasing important to develop architectures that are resilient to variation.

One visible implication of parameter variation is that microarchitectural structures will have considerable *manufacturing-induced asymmetry*. On-chip resources which are designed to be identical with respect to power and performance, may have significantly different characteristics when fabricated. This applies to portions of caches, functional units, and other microarchitectural structures within and across cores. In this paper, we focus on *manufacturing power asymmetry* which as opposed to architected power asymmetry [20], cannot be managed at design time to produce a balanced design that offers performance and power benefits. In contrast, differences in asymmetric resources must be discerned in a post manufacture step and any resource balancing policies must be customized to fit a specific chip.

For a chip multiprocessor (CMP) under power asymmetry, there are two important types of policy decisions. First, for a single thread running on a core, there is a choice of which resources should be used and which resources can be transitioned to lower power standby modes to achieve a within-core power/performance balance. This decision may involve not just what capacity to choose for a resizable cache, but also which subarrays of the cache should be chosen. Second, for many threads running on a CMP, there is a global choice of which threads to assign to which cores. For example, an integer application which does not use floating point resources might be suitably run on an execution core with a leaky floating point multiplier because it could be power-gated without harming performance. On the other hand, a floating point application would be a poor choice because there are fewer power-gating opportunities and hence the high leakage factor factors prominently. By nature, these decisions must consider the resource requirements of individual applications versus the capability of each core to satisfy those demands.

This paper explores mechanisms and policies to match physical resources to usage opportunities to achieve power/performance benefits. Specifically, we make the following primary contributions:

- We propose a novel resizable cache organization that takes advantage of spatial patterns in leakage variation to reduce static power.
- We evaluate the benefits of matching threads to physical cores on a chip-custom basis to reduce leakage power under extreme parameter variation.

The rest of this paper is organized as follows: In Section 2, we discuss the impact that manufacturing variations can have on power asymmetry. In Section 3, we introduce techniques for matching application utilization patterns to physical resources within a core and across a chip multiprocessor. These strategies allow us to apply aggressive power optimizations to improve energy-profiles with minimal impact on performance. In Section 4 and 5, we present experimental methodology and evaluate the benefits of resource matching within and across CMP cores. In Section 6, we present a discussion and comparison to previous work. Finally, we offer conclusions in Section 7.

2 Background

In high-performance chip fabrication, manufacturing errors introduce physical deviations in circuit structures. As a result, gates and interconnects have physical and electrical parameters which differ from their intended values in both random and systematic fashions. These errors are the result of technology scaling, limitations in fabrication tools/processes, and random chance. Together they can have a detrimental impact on the performance, power, and reliability of circuits. Technology forecasts predict that parameter variations will continue to be a problem in deep submicron design for the foreseeable future [29].

Some of the most problematic types of manufacturing error can contribute to variance of the effective v_t, a critical power and performance parameter for transistors. Specifically, variations in gate length, gate oxide thickness, and channel dopant concentration can all affect v_t. In particular, higher values of v_t produce transistors that switch at a slow rate, limiting clock frequency. On the other hand, lower values of v_t create transistors that consume considerable amounts of static power due to an exponential relationship between v_t and leakage current. Leakage is the primary source of power variability in current high-performance processors [4]. Due to rising relative variation in manufacturing error [29], the emergence of power as a first order design constraint [24], and the growing prominence of static power [6, 11, 12, 28], leakage variability will be an important topic for years to come. In this paper, we focus on the leakage power introduced by manufacturing variation.

The statistical distribution of process parameters has a significant impact on leakage. Previous work has shown that gate length in particular, shows strong spatial correlation patterns [13]. Consequently, local clusters with leaky transistors can appear across a chip. As a result, structures at the microarchitectural scale could have very different leakage factors depending on where these clusters form, how dense they are, and how far they spread. While circuit techniques such as adaptive body-bias [34], can help to mitigate gate length variation by effectively offsetting v_t, these approaches may not be suitable for fine-grain parameter correction under extreme within-die parameter variation. In particular, local adaptive body bias requires an expensive triple well process and bias voltage generation at local sites. Solutions based solely on circuit techniques may not offer the best tradeoffs in future high-performance processors. We investigate architectural techniques that can complement circuit focused approaches for mitigating variability.

3 Matching Asymmetric Physical Resources

The notion that different processor cores can be designed to meet different performance and power objectives is one of the key motivations behind heterogenous multicore architectures [20]. Under this design paradigm, architects can artfully combine processor cores that have different intrinsic microarchitectural structures. This diversity creates dissimilar performance capabilities and offers different design points on the power/performance continuum. In terms of area efficiency, this can produce significant performance benefits for a fixed amount of silicon real estate [21]. With respect to individual applications, heterogeneity and thread migration allow a CMP to run code on low power cores when program phases allow and transition to high-performance cores when necessary [20]. For multiprogrammed workloads, heterogeneity allows the processor to benefit from higher throughput per unit area and match threads to suitable cores when there is sufficient thread diversity. However, the effects of manufacturing induced leakage variation can upset the power/performance balance of heterogenous CMP designs and in some cases introduce power asymmetries in what would otherwise be homogenous CMPs.

3.1 Asymmetric Resource Thread Matching: Coarse Granularity Variability Management

In this work, we introduce Manufacture Aware Thread Matching (MATM), a simple strategy for multithread assignment that makes the best of chip variation and assigns threads to cores to minimize power consumption without compromising performance. The basic tenant behind MATM is that on a chip-to-chip basis, physical cores will have different power profiles as a result of manufacturing variation. By appropriately matching threads to cores, we can maximize the benefit of existing power saving strategies such as power-gating of idle resources. For the purposes of this discussion, we focus on power asymmetries that appear in homogenous CMPs. However, we note that this strategy is clearly applicable to heterogenous CMPs which may have their power/performance tradeoffs upset by manufacturing variation.

Software Support

Traditionally, system software is responsible for assigning software threads to execution hardware to promote throughput, turn-around time, and fairness, among other criteria. These scheduling decisions may consider data locality, communication patterns, performance capabilities of the hardware, as well as thermal and energy properties of the processor. However, *identically designed hardware resources are thought to be truly identical to the system software*. Consequently, traditional schedulers cannot account for manufacturing variation.

Under MATM the system scheduling software is made aware of the individual power profiles of individual caches, cores, and other resources. With knowledge of a power profile and thread resource usage patterns, the system software calculates the effective energy savings of each thread on every core. It then chooses an assignment of threads to cores that allows microarchitectural power management strategies at the core-level to reap their full potential by choosing thread assignments that lead to the most advantageous power-gating opportunities. Given a prediction on what power savings each thread will see on each core based on leakage factors and resource requirements, MATM-capable management software can find an optimal assignment in polynomial time. In general, the thread assignment which produces the best global power savings for n cores can be identified in $O(n^3)$ time by the Kuhn-Munkres Algorithm, known commonly as the Hungarian Method [25, 19]. For current designs, when n is relatively small, this a reasonable computational overhead. In future designs where, n is large, a more efficient algorithm which generates an approximate solution may be required. We reserve further exploration of this topic for future work. Resource power profile information is by nature specific to a physical chip instance. In particular, this work assumes that the leakage variation can be detected at the level of microarchitectural blocks as part of a post fabrication step. We outline a simple strategy for this in Section 3.3.

Operating systems kernels serve as the traditional vehicle for resource scheduling in multiprocessor environments and could be augmented to support MATM. However, the implementation specific details of power variation and dynamic hardware configurations may make this kernel-level solutions less desirable because many details are exposed to the ISA. An alternative is to instead implement MATM within a microvisor, a light-weight virtual machine monitor (VMM) that is co-designed with the hardware. With this approach, the microvisor executes a superset of the standard ISA at the highest privilege level. The additions may include implementation specific details such as instructions which read leakage profile registers. In this way, the conventional application and operating systems software can benefit from MATM without code modification.

Previous work has examined profiling and tuning techniques that can be used by systems software to identify resource usage patterns [9, 10]. This information is used to guide microarchitectural optimizations such as cache resizing and could be made available to management software to determine how much resizing will occur for each thread. By combining resource usage information and power profiles MATM-capable management software can project the power for each thread on each core.

Power Saving Mechanisms and Power/Performance Tradeoffs

On a homogenous multicore architecture under variability, the potential of MATM is determined by the available microarchitectural power saving techniques. Fine-granularity energy conscientious strategies such as power-gating are particularly well-suited for mitigating static power, the primary source of power variability. In particular, the resizing of architectural structures such as caches and issue queues provide an opportunity to apply power-gating to unused subarrays and entries [1, 7, 11, 15, 35]. Due to the increasing prevalence of leakage power in high-performance designs, these strategies are likely to gain wider acceptance.

Figure 1 presents an illustrative example of how resource matching can improve the energy profile of a workload. This example considers a two thread workload which must be mapped onto a dual core homoge-

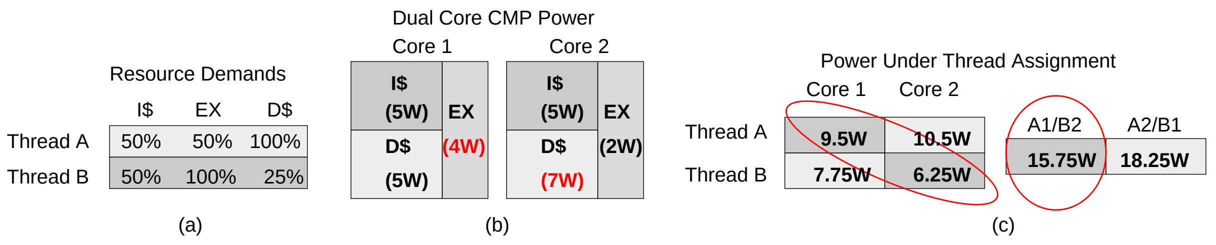

Figure 1: An example showing how resource matching can reduce power consumption in multicore architectures with manufacturing induced asymmetry. The data cache of Core 2 consumes more power than that of Core 1 while the execution units of Core 1 consume more power than those of Core 2. By noting that Thread A and B have different resource demands with respect to these components, a considerable savings can be achieved by assigning Thread A to Core 1 and Thread B to Core 2.

nous CMP under power variability. Both threads need at least 50% of the instruction cache to meet their performance requirements. However, the two threads have very different execution unit and data cache requirements. We see that the physical cores also have mismatches in their component power values. As a result of the two possible thread assignments: $\frac{T_A \rightarrow C_1}{T_B \rightarrow C_2}$ and $\frac{T_A \rightarrow C_2}{T_B \rightarrow C_1}$ can produce different total power results. The largest benefit comes from assigning Thread B to Core 2 where it benefits from the lower execution unit power. This is crucial for Thread B because it places heavy utilization demands on this portion of the processor. There are no effective savings for Thread A under either assignment, so Thread B's preferences dominate and the $\frac{T_A \rightarrow C_1}{T_B \rightarrow C_2}$ produces the best power savings.

In general, MATM can be applied under different combinations of core-level optimizations. This allows for a rich set of power/performance tradeoffs. For instance, adaptive microarchitectures commonly resize caches to meet a specific allowable miss rate or percentage IPC degradation over a baseline. Because MATM can achieve a lower power cost at the same given performance level as a variation unaware assignment policy, more aggressive performance targets can be set for a fixed power limit. Power variability also introduces interesting challenges for optimizing unified power/performance metrics like $energy\_delay$ and $energy\_delay^2$. Because cores have unique power profiles, different microarchitectural configurations might produce optimal $energy\_delay^N$ values for the same application running on different cores. Hence the choice of what configuration to use for a thread is also dependent on which core it is running on. This is a sharp contrast from performance centric configuration policies which would make the same choice regardless of which core a thread is assigned to.

In all, manufacture aware thread matching (MATM) offers some hope in limiting the effects of power variability in multicore architectures. With minimal support from system software, threads can be assigned to cores that yield the most benefit from power saving strategies.

3.2 Prioritized Cache Sets: Fine Granularity Variability Management

In addition to the considerable benefits of matching threads to cores in a variability sensitive processor, finer granularity resource matching can improve the efficiency of a local processor core or cache. In particular, a number of dynamic sizing strategies have been proposed for energy reduction of fetch queues, issue queues, and caches [1, 7, 35]. In essence, these approaches identify the minimum number of entries necessary to support performance goals for a given application. This can be done through either off-line analysis or on-line analysis. The resources are then resized by disabling unnecessary portions of these structures, yielding improvements in dynamic and static power with limited performance impact. However, these approaches all assume that all portions of these resources are identical in nature. They focus on meeting a targeted aggregate entry count and ignore manufacturing asymmetries.

In particular, cache arrays are an example of large, regular structures that may be prone to local parameter variations. If a variability agnostic sizing policy chose to resize such a cache, it could unwittingly choose to enable a subarray which had a high leakage factor. On average this could amount to an appreciable deviation, and in extreme cases the differences might be severe. By allocating portions of its data array judiciously, a variation aware processor core stands to improve its energy profile.

Resizing with Sets

Previous work on resizable cache arrays has focused on both extremely fine granularity enabling/disabling for cache blocks [18, 27] as well as commissioning/decommissioning of subarrays [1, 35]. In the former, cache lines may be power gated to save leakage energy when their data is not in use. When this occurs, the *active size* (instantaneous capacity) of the cache changes as a result of changing demand. Under subarray resizing, the cache's active size is chosen to meet a target performance goal. These organizations may save both static and dynamic power by power gating enough subarrays to achieve the desired active size and disabling all precharges to cells in the de-commissioned subarrays. In this work, we focus on resizing strategies at the subarray level because they have more modest overhead requirements on power gating circuitry.

Resizing at the cache way level was first proposed by Albonesi [1], and has recently been used as a basis for simple variation aware cache sizing [23]. The original concept, *selective cache ways*, first divides a cache into subarrays that correspond to individual cache ways. To meet the active size target, ways may be disabled, simultaneously decreasing associativity and capacity. This allows for an efficient, low-overhead implementation. When the cache is resized, dirty blocks in affected subarrays are first written back to the next level of the memory hierarchy. When all dirty blocks have been written back, the power or ground for the affected subarrays are gated, decreasing their leakage power to effectively zero. On successive cache accesses, the wordline and bitlines to disabled ways are not longer pre-charged. The data selection logic ignores disabled cache ways and only performs tag-matches for enabled ways.

Prioritized cache ways builds on selective cache ways by choosing **which ways** to enable based on their leakage profiles [23]. Specifically, a pre-initialized leakage priority register keeps track of the leakage factor for each subarray. As the cache is resized, the hardware reads the priority register to choose which subarrays to disable. Initialization strategies for the priority registers and strategies for identifying leakage factors were discussed in [23]; we revisit them in Section 3.3. With the appropriate prioritization, the hardware can choose a subset of ways that yield the best energy results for a given active size. A major limitation of selective and prioritized cache ways are that they decrease associativity as they decrease capacity, possibly introducing conflict misses. In addition, sizings are limited by associativity, preventing these approaches from being used on low-latency direct mapped caches and hindering efficacy on low associativity caches.

An alternative that we consider in this work is to resize by choosing the **number of sets** that the cache will support. This concept, known as *selective cache sets*, was proposed by Powell et al [35]. In essence, this approach is suitable for caches with low associativity and does not introduce conflict misses when resizes. It does however require some modification to tag arrays and data array indexing. Because selective sets change the mapping of effective addresses to cache sets, resizes force all dirty cache lines to be written back to the next level of the memory hierarchy. However, resizing is an infrequent event, so the cost of writeback can be minimized.

Under selective sets, the cache may be downsized from its maximum capacity using powers of two as intermediate sizes. The tag array is extended to accommodate tags that would be used under the cache's minimal size. For example, a 64KB cache that could be resized to a minimum of 8KB would require three extra tag bits. As the cache is resized, the tag comparison logic tracks the current size for the cache and uses the appropriate number of tag bits. In addition, when the cache is downsized, it uses fewer bits in set indexing. The *set-mask*, a simple shift register, helps to calculate index bits under resizing by tracking the correct bitwidth for indexing. On a cache access, the effective address is Boolean ANDed with the set-mask to produce the correct set index. As the cache is decreased, the set-mask is shifted to the right, introducing leading zeros and effectively decreasing the number of bits used in the set-index operation. Under this scheme, the highest numbered sets in powers of two are *always* disabled when the cache is downsized.

A New Cache Organization: Prioritizing Sets

Drawing on the observation that selective sets always disable specific subarrays, we introduce *prioritized cache set* resizing that exploits choice to reduce leakage power under variability. Similar to selective sets, our approach allows the cache to be appropriately sized by introducing a mask register that reduces the number of index bits. The additional goal of prioritized cache sets is to allow the mapping to vary on a core-to-core/chip-to-chip basis to produce the largest possible savings on cache energy. To meet that requirement, we introduce the *set-remap* register which is used to select which subarrays to use for a given resize capacity. As shown in Figure 2, effective addresses are first Boolean ANDed with the set-mask register as in selective sets to reduce the number of index bits. Then they are Boolean ORed with the value held in the set-remap register to remap the indices to a specific region of the cache array. As shown in Figure 2, the map table holds the corresponding set-remap values to use under various sizing configurations. We assume that these entries can be populated based on independent post-manufacture leakage measurements for each cache on each chip in the same fashion that the leakage priority register is populated for prioritized cache ways. With the exception of the additional remap step, general operation of the cache proceeds as in does under selective cache sets. Specifically, sizing choices are limited to powers of two and resizings require that all dirty blocks be written back.

The remap strategy uses a very simplistic approach which should not have a negative impact on cycle time. We choose to perform a simple Boolean OR operation to apply the choice of remapped subarray. Because this is a simple bitwise operation with potentially early arriving values read from a register, we do not feel that the remap stage adds an appreciable overhead to set indexing. This simple design does however have one slight drawback: the active cache will always be chosen from contiguous set indices. For example, if the cache were downsized to 16KB, instead of being able to choose the two best 8KB subarrays which may be non-contiguous, we would have to settle on a single, contiguous 16KB subarray. An alternative would be to use an independent table that would allow for an unrestricted remapping that includes non-contiguous cache components. This would likely add extra delay overhead that would increase cycle time hence we do not consider such an organization in this paper. We perform studies in Section 5 that show that the lost energy benefits are small and the simple ORed set mapping works well in practice.

Under prioritized sets, the resizing decision can be made either off-line during compiler analysis or on-line by profiling hardware. Resizing policies have been studied extensively in previous work [10, 35]. The major improvement of prioritized sets is the choice of *which sets to enable*, not how many sets to enable. Consequently, we focus on those implications in our evaluations and discussions.

3.3 Detecting Power Asymmetry

The measurements needed to identify asymmetry can be collected off-line during the manufacturing test phase or during a cold system boot. Individual processor components 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 component can be calculated from ammeter readings of total chip current draw or in-field power sensors. Collected data can be quantized, physical structures can be sorted by their leakage power, and the resultant information can be kept in non-volatile near-chip storage. To reduce boot time overhead, registers and tables can be configured based on previously determined values held in non-volatile memory.

4 Experimental Methodology

Our experiments model the performance of a 4-core homogenous chip multiprocessors for a 65nm process. Each core of the processor is comparable to an Alpha 21264 (EV6) scaled to current technology [14]. Under this simple technology scaling, we assume that the processor will not be able to reach the maximum frequency for 65nm, and instead operates at a 3.0GHz frequency. A similar scaling methodology was used by Kumar et al. in [20]. The cores in the processor have private L1 data and instruction caches and private L2 unified caches. Table 1 summarizes our base processor model.

Our simulation infrastructure is based on a heavily modified version of the M5 Stand-Alone Execution simulator [3] which includes detailed models of pipelines, caches, buses, and off-chip memory.

Single Core	
Feature Size	65nm
Clock Rate	3.0 GHz
Fetch/Decode Width	4 inst
Issue Width	6 inst, out-of-order
IQ/LSQ/ROB	32/40/80 entries
Functional Units	4 IntALU, 1 IntMult/Div
	1 FPALU, 1 FPMul/Div
	2 MemPorts
L1 Inst Cache	64KB 2-way 64B blocks
L1 Data Cache	64KB 2-way 64B blocks
	3 cycle load hit
Chip Multiprocessor	
Cores	4
L2	2MB 16-way shared 128B blocks
Off-chip memory latency	200 cycles

Table 1: Processor Parameters

Our approach to modeling on-chip leakage power variation is similar to that of [23], but we also include macro models for different logic/memory structures in the pipeline. In essence, we perform Monte Carlo experiments which model spatially correlated gate width and gate length variation. For each Monte Carlo sample, we generate 1024×1024 grids which model Gaussian distributed local gate length and gate width variation. Over these grids, the gate parameters are correlated as a function of distance as in [13]. Leakage power is modeled using a combination of curve fit and tabulated projections based on SPICE-level simulation for subthreshold leakage. [28]. In our experiments, we assume a 3σ variation of 9% and we perform 10,000 Monte Carlo trials.

To evaluate the efficacy of our management approach, we use several workloads that showcase a variety of usage patterns which would be likely in a multiprogrammed environment. Individual applications are taken from the SPEC CPU2000 benchmark suite. To re-

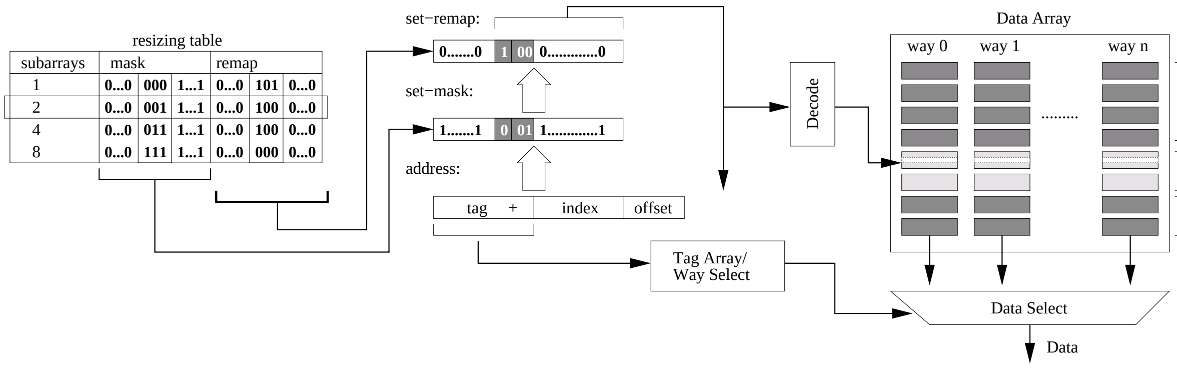

Figure 2: Hardware organization for prioritized cache sets. Novel components include the set-remap register which applies the choice of prioritized subarrays to the set index operation. In this example, the cache has been resized to use two subarrays. The set-remap register chooses the 4th and 5th subarrays (10*).

duce the total number of simulations, we identify a subset of SPEC applications which exhibit a range of behaviors. To isolate representative simulation windows, we use SimPoint execution intervals [30].

5 Results

5.1 Prioritizing Sets in Resizable Caches

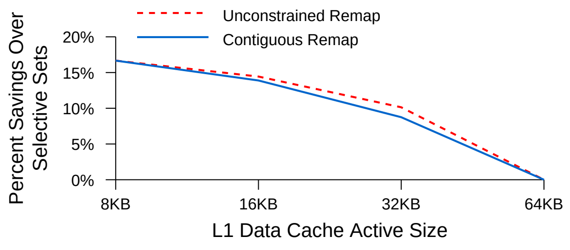

Figure 3: Static power savings versus unaware selective cache sets for various active cache sizes.

Figure 3 shows the average 64KB L1 cache static power savings that variation-aware set resizing can potentially achieve over variation-insensitive resizing. The expected savings at the 8KB minimal size are around 17%, and these savings gradually decrease as the active size increases. Figure 3 also shows that the simple contiguous remapping policy proposed in Section 3 is very competitive with a more aggressive unconstrained remapping policy. The agressive policy which we call "Unconstrained Remap" allows the hardware to use any 8KB subarrays which consume the least leakage power. This would likely require significant hardware overhead to perform all the possible remaps. On the otherhand, simple contiguous remap policy can use simple bit-wise AND and OR operations, reducing hardware overhead with almost the same leakage savings. This is due to spatial correlation in the transistors dimensions. Neighboring subarrays are likely to have similar leakage factors, providing little benefit for unconstrained remapping.

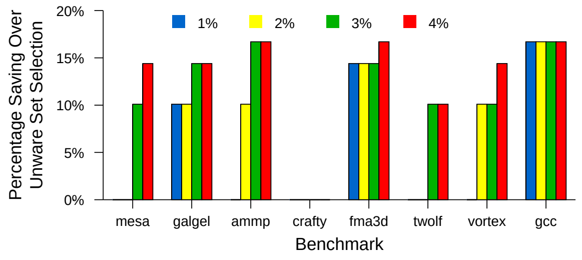

Figure 4: Static power savings versus unaware selective cache sets for representative benchmarks with variant performance loss tolerance.

In Figure 4, we present relative L1 data cache savings for a representative subset of the SPEC2000 benchmarks under different performance tolerances. In all of our experiments, we assume static resource sizing. In general, benchmarks that place less stress on the L1 cache get larger relative benefits from prioritized set sizing. Benchmarks such as mesa and ammp have different L1 sizing choices as the allowed performance degradation varies from $1-4\%$. As a result, they can achieve a wide range of benefits. Other benchmarks like crafty and gcc do not allow for much L1 sizing and see rather fixed benefits under different performance tolerances.

To summarize our findings across the benchmarks, we plot the average power savings from prioritized sets under different performance tolerances in Figure 5. The solid line shows the mean over the full SPEC2000 suite. An average of 8% to 12% savings is possible when the

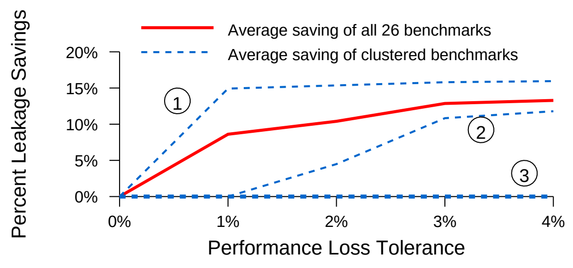

Figure 5: Static L1 Icache power savings versus unaware selective cache sets from the average of all 26 benchmarks and clustered benchmarks under k-means grouping.

tolerable loss moves from 1% to 4%. We further divide the benchmarks into distinct subgroups which have similar resizing potential using k-means clustering. The three dashed lines in Figure 5 illustrate the average savings of the individual groups. The top line, which represents the largest group (15 benchmarks), indicates that large savings can be achieved for as little as a 1% performance loss. The bottom dashed line, which overlaps the x-axis of the graph, represents the few (2) benchmarks that cannot endure any L1 D-cache downsizing under the given performance loss tolerance. The middle dashed line represents the remaining (9) benchmarks which only show significant benefit when the performance tolerance increases to 3%.

5.2 Resource Matching Within Cores

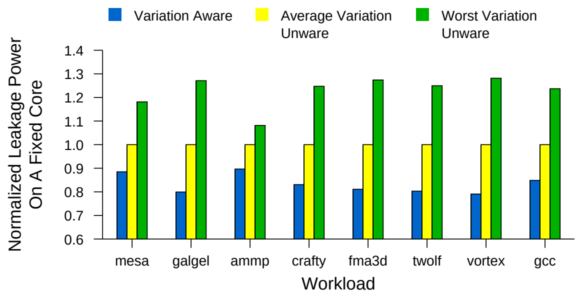

Figure 6: Static power of representative benchmarks tolerating 2% performance loss. Results are normalized with respect to a baseline situation in which units are randomly selected for power-gating, ignoring variation.

In this section, we study the potential of static power savings through variation-aware resource matching in a single core. The same eight benchmarks depicted in Figure 4 are chosen for exploration here. We tolerate a maximum 2% performance loss for each thread and apply: selective cache way resizing for L2 caches, selective cache set resizing for L1 caches, and functional unit power gating during long idle periods [15].

Figure 6 shows the normalized leakage savings. The center bars represent the baseline average in which closed microarchitecture blocks are randomly selected without knowledge of power asymmetry. The results are normalized to this value for every application. The bars on the left show the static power usage when variation-aware block selection is made. The expected savings over the baseline ranges from 10% to 21%, with an average of 17%. The bars on the right correspond to the most unfortunate situation, a pathologically bad case were the most power-hungry blocks are chosen for service. Under this worst case, maximal power is consumed – 8%-28% more than the baseline.

5.3 Resource Matching Across Cores

To evaluate the performance benefits of resource matching for multiple threads on multiple cores, we first constructed four workload mixes out of SPEC2000 integer and floating-point benchmarks. These workloads are representative of a codes that might appear in multiprogrammed environment where there may be varying amounts of thread-level diversity. We present the potential benefits of MATM policies on these workloads in Figure 7.

Thread matching policies can have a range of energy benefits for the different workload mixes. Overall, the mesa-galgel-ammmp-crafty mix benefits the most from variability conscious thread assignment, with a 16% reduction in normalized leakage energy over unaware assignment in the average case. This is because this mix has the greatest thread diversity in terms of resource requirements. In particular, ammp uses most of its 2MB L2 cache and cannot tolerate resizing. The other benchmarks place considerably less demands on L2 size and can hence operate with a downsized cache. As a result, if ammp is placed on a core which has a leaky L2 cache, it will not downsize due to the performance constraint. The power impact can be significant as witnessed by the worst-case unaware assignment which has a 20% increase over the average-case normalized leakage. MATM will avoid this by placing ammp on a core which has a low leakage factor for its private L2. At the opposite extreme, the vortex-gcc-mesa-galgel mix benefits the least from resource matching with an appreciable, but more most 7.5% reduction in normalized leakage. In this case, the benchmarks all place comparable resource demands on their prospective processor cores. As a result, there is less of an impact for a poor assignment decision.

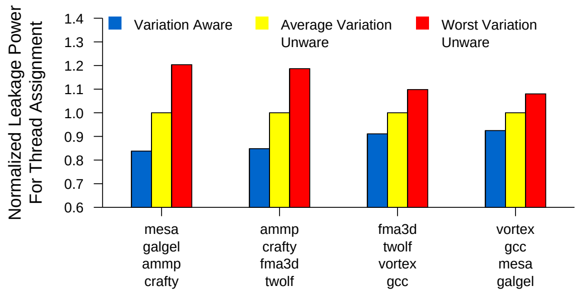

Figure 7: Average static power of whole chip for thread to core assignment normalized to baseline where the assignments are made at random. All programs use resizing policies that tolerate a 2% performance loss.

Unit	ammp	crafty	fma3d	twolf	chip
FU	0.80	0.97	1.20	1.00	1.00
L1 I/D	0.77	1.02	1.20	0.96	0.91
L2	0.79	0.73	1.37	1.04	0.88

Table 2: Breakdown of leakage power savings on power-gated structures. Results are normalized to per unit static power under random thread assignment. Within a core, we apply variation aware power-gating.

In Table 2 we present normalized leakage savings for different microarchitectural structures for the ammp-crafty-fma3d-twolf workload mix. We can see that ammp gets significant benefits for all of its structures. This is because ammp places the largest size demands on L2. This virtually guarantees that it will run on the least leaky core. It benefits with respect to L1 and FU power due to spatial correlation. We also see that some of the benchmarks, most notably fma3d actually increase in some categories. This is because one benchmark (ammp) always gets the least leaky core and one benchmark has to suffer with the leakiest core (fma3d). Despite this, the overall savings are still good.

6 Discussion

While parameter variation has received considerable attention in the electronic design automation and circuit communities, it is a relatively new topic for architects as of this writing. Previous work at the architecture-level has developed models for power and performance due to Process, Voltage, and Temperature (PVT) variations [16] and examined the influence of variability on multi-core chips [17]. The relationship between architecture/circuit design choices and variability was explored in [2]. In [22], the authors dicuss the impact of uncertainty on microarchitectural design and identify benefits of using GALS architectures as one way to counteract parameter variation.

Our work is the first to consider the relationship between scheduling/assignment decisions made in systems software to low-level parameter variation. We believe that our approach can be complimentary to microarchitecture and circuit focused strategies because it can optimize globally, across the chip. As challenges related to parameter variation mount, architects will increasingly have to look for solutions that span many levels of the design space and management hierarchy.

7 Conclusion

In this work, we present Manufacture Aware Thread Matching (MATM), a strategy for thread assignment that reduces the overall power consumption for chip multiprocessors fabricated under extreme parameter variation. By matching thread behaviors to power profiles for hardware structures, MATM can find an optimal, customized thread assignment for a chip. The key insight is that existing microarchitectural power reduction techniques will have different degrees of success on the cross product of threads and processor cores. With knowledge of the leakage power of its host chip, system software can identify the right assignments to maximize the benefit of core-level adaptation and minimize total leakage over the entire chip. On average, our approach nets a 12% reduction in normalized leakage power for the workload mixes that we evaluate.

We also introduce a cache organization that reduces the impact of leakage variation by exploiting choice in subarray selection. Because it operates on cache sets rather than cache ways, it can be implemented in low associativity caches used in L1 instruction and data caches. Our approach can reduce the L1 leakage power by 10% over previously proposed cache resizing strategies that do not consider parameter variation.

References

[1] D. H. Albonesi. Selective cache ways: On-demand cache resource allocation. In *International Symposium on Microarchitecture*, pages 248–, 1999.

[2] N. S. K. an T. Kgil, K. Bowman, V. De, and T. Mudge. Total power-optimal pipelining and parallel processing under process variations in nanometer technology. In *Proc. Int. Conf. of Computer Aided Design (ICCAD-2005)*, November 2005.

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

[4] S. Borkar et al. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th Design Automation Conference (DAC-40)*, 2003.

[5] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA-7)*, January 2001.

[6] J. A. Butts and G. S. Sohi. A static power model for architects. In *Proceedings of the 33rd International Symposium on Microarchitecture (MICRO-33)*, December 2000.

[7] A. Buyuktosunoglu, T. Karkhanis, D. H. Albonesi, and P. Bose. Energy efficient co-adaptive instruction fetch and issue. In *Proceedings of 30th International Symposium on Computer Architecture (ISCA-30)*, May 2003.

[8] 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.

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

[10] A. Dhodapkar and J. E. Smith. Tuning recongurable microarchitectures for power efficiency. In *The 11th Recongurable Architectures Workshop (RAW 2004), held in conjunction with the 18thInternational Parallel and Distributed Processing Symposium*, April 2004.

[11] S. Dropsho, V. Kursun, D. Albonesi, S. Dwarkadas, and E. Friedman. Managing static leakage energy in microprocessor functional units. In *The 35th International Symposium on Microarchitecture (MICRO-35)*, 2002.

[12] 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.

[13] 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.

[14] L. Gwennap. Digital 21264 sets new standard. *Microprocessor Report*, pages 11–16, Oct. 28, 1996.

[15] Z. Hu et al. Microarchitectural techniques for power gating of execution units. In *The International Symposium on Low Power Electronics and Design (ISLPED)*, July 2004.

[16] E. Humenay, W. Huang, M. R. Stan, and K. Skadron. Toward an architectural treatment of parameter variations. Technical report.

[17] E. Humenay, D. Tarjan, and K. Skadron. Impact of parameter variations on multi-core chips. In *Proceedings of the 2006 Workshop on Architectural Support for Gigascale Integration, in conjunction with the 33rd International Symposium on Computer Architecture (ISCA)*, June 2006.

[18] 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.

[19] H. W. Kuhn. The hungarian method for the assignment problem. 2, 1955.

[20] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-isa heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the 36th International Symposium on Microarchitecture (MICRO-36)*, December 2003.

[21] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st International Symposium on Computer Architecture (ISCA-31)*, June 2004.

[22] D. Marculescu and E. Talpes. Energy awareness and uncertainty in microarchitecture-level design. *IEEE Micro*, 25:64–76, Sept.-Oct. 2005.

[23] K. Meng and R. Joseph. Process variation aware cache leakage management. In *Proceedings of the International Symposium on Low-Power Electronics and Design (ISLPED)*, October 2006.

[24] T. Mudge. Power: A first class design constraint. *Computer*, 34:52–57, April 2001.

[25] J. Munkres. Algorithms for the assignment and transportation problems. 5:32–38, March 1957.

[26] S. R. Nassif. Modeling and forecasting of manufacturing variations. In *Proceeding of the Fifth International Workshop Statistical Metrology*, 2000.

[27] M. Powell et al. Gated-vdd: A circuit technique to reduce leakage in cache memories. In *In Proceedings of Intl. Symposium on Low Power Electronics and Design*, July 2000.

[28] 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.

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

[30] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceedings Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct 2002.

[31] D. Singh and V. Tiwari. Power challenges in the internet world. In *Cool Chips Tutorial at (MICRO-32)*, Nov 1999.

[32] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal RC-modeling for accurate and localized dynamic thermal management. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA-8)*, February 2002.

[33] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proceedings of the 30th International Symposium on Computer Architecture (ISCA-30)*, June 2003.

[34] J. W. Tschanz et al. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. 37(11), November 2002.

[35] S.-H. Yang, M. Powell, B. Falsafi, K. Roy, , and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Seventh International Symposium on High-Performance Computer Architecture (HPCA-7)*, January 2001.