Variation-Tolerant Non-Uniform 3D Cache Management in Die Stacked Multicore Processor

Bo Zhao, Yu Du‡, Youtao Zhang‡, Jun Yang
Electrical and Computer Engineering Department
‡Department of Computer Science
University of Pittsburgh, Pittsburgh, PA 15261
{boz6, juy9}@pitt.edu, ‡{fisherdu, zhangyt}@cs.pitt.edu

ABSTRACT

Process variations in integrated circuits have significant impact on their performance, leakage and stability. This is particularly evident in large, regular and dense structures such as DRAMs. DRAMs are built using minimized transistors with presumably uniform speed in an organized array structure. Process variation can introduce latency disparity among different memory arrays. With the proliferation of 3D stacking technology, DRAMs become a favorable choice for stacking on top of a multicore processor as a last level cache for large capacity, high bandwidth, and low power. Hence, variations in bank speed creates a unique problem of non-uniform cache accesses in 3D space.

In this paper, we investigate cache management techniques for tolerating process variation in a 3D DRAM stacked onto a multicore processor. We modeled the process variation in a 4-layer DRAM memory to characterize the latency variations among different banks. As a result, the notion of fast and slow banks from the core's standpoint is no longer associated with their physical distances with the banks. They are determined by the different bank latencies due to process variation. We develop cache migration schemes that utilizes fast banks while limiting the cost due to migration. Our experiments show that there is a great performance benefit in exploiting fast memory banks through migration. On average, a variation-aware management can improve the performance of a workload over the baseline (where one of the slowest bank speed is assumed for all banks) by 17.8%. We are also only 0.45% away in performance from an ideal memory where no process variation is present.

Categories and Subject Descriptors

B.3 [**Hardware**]: Memory Structures

General Terms

Design

Keywords

Process Variation, 3D Die Stacking, DRAM, NUCA

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MICRO'09, December 12–16, 2009, New York, NY, USA.
Copyright 2009 ACM 978-1-60558-798-1/09/12 ...$10.00.

1. INTRODUCTION

Scaling and integration for nanoscale CMOS transistors and circuits present promising device density and performance trends. However, one of the major hurdles in scaling in nanometer regime is the difficulty in controlling the device characteristics precisely during fabrication, which leads to *parameter* or *process variations*. Key characteristics in both devices and wires affected by process variations are either structural, such as gate length, wire width etc., or electrical, such as threshold voltage, resistance per unit length, etc. [29]. Those parameter variations have significant impact on circuit performance, leakage and reliability. We will focus on their impact on circuit performance in this paper.

Process variations can be classified into within-die (WID) variations and did-to-die (D2D) variations. WID variations refers to the differences in device features among transistors on one die, while D2D variations refers to such mismatches among different die. There have been active efforts recently on WID variation tolerant designs such as improving die yield [1, 30], improving cache designs [11, 21], improving reliability [13, 14, 33], mitigating latency variations in pipeline stages or various components in a chip [9, 20, 22, 40], and lowering energy consumption or improving energy delay product [15, 38]. Recently, the proliferation of three-dimensional (3D) die-stacked architecture gives rise to considerations in both WID and D2D variations. 3D stacking is a technology that stacks multiple active silicon die on top of each other, and connect them through wafer bonding. The communication among different layers is carried in Through Silicon Vias (TSVs). With this technology, it is possible to either stack multiple 2D die into a powerful 3D chip, or implement a processor using true 3D circuits. In either design choice, process variations impact both the performance within each layer and the communication among different die. That is to say, the performance of the overall 3D processor is constrained by both WID and D2D process variations.

Among various 3D architectures, stacking cache or memory chips directly on top of a 2D multicore chip [4, 17, 23–26, 39] has gained its popularity due to 1) immediate boost in on-chip memory capacity without increase in die area; 2) short core-to-memory interconnect delay because data can be supplied vertically through TSVs which are orders of magnitude shorter than horizontal wires that connect core with memory within a die; 3) best heat dissipation capability because the most active core layer can be built closest to the heat sink; and 4) good scalability in number of layers. A recent work performed a comprehensive study on performance-energy tradeoffs between stacking SRAM and DRAM on top of a 2D multicore layer. The conclusion is that using commodity DRAM cells with low standby power peripheral circuitry generates the best energy-delay product [39]. This is

primarily due to the low power and high density of DRAM arrays which can compensate its low speed with large capacity (and hence low miss rates) within the same die area, when compared with SRAM arrays. There have been great amount of efforts on on-chip SRAM cache optimizations under processor variations [1, 2, 11, 18, 21, 30, 35]. However, there is little work on DRAM variations probably because DRAMs are conventionally off-chip, so their latency changes due to process variation have little impact on the overall performance. Previous research showed that there can be 18ns read access latency (t_{RAC}) variation in main memory DRAM under process variation [42]. With the 3D integration of DRAM chips with multi-core chips, such latency variations will become more pronounced, especially where there are both WID and D2D variations in a multi-layered DRAM stack.

In this paper, we first model the process variations in a multi-layered 3D DRAM stack integrated with a multicore processor, then develop memory management techniques to overcome the process variations. Our 3D DRAM architecture follows closely a commercial product from Tezzaron Corporation [46,47]. From our modeling results, we observed that the DRAM subbanks in 3D space present a fairly wide range of access latencies due to process variations. Such latency variations create a 3D non-uniform access memory for each core on-chip. Similar to the NUCA problem in a 2D CMP, 3D NUMA (or NUCA, if the DRAM is used as a cache) has a significant impact on the performance of the entire chip. Unlike the 2D NUCA problem, such non-uniformity is dominated by process variations, and less due to the wire delay or interconnection network delay as in a 2D CMP. We develop cache/memory management techniques to overcome the non-uniform access times from different memory banks. The basic idea is to migrate data from slow banks to fast banks to reduce the average DRAM access time. Such migration was shown by our experiments to be very effective, especially when the working set of a workload fits into the fast banks due to the large capacity provided by DRAM. In those cases, we even achieve performance improvements over the ideal chip where there is no process variations. We further devised our migration schemes such that the energy increase is minimized. On average, our variation-tolerant migration scheme is 17.8% better than a conservative chip where the speed of the slowest bank is used as the nominal speed for all banks, and only 0.45% away from the performance of the ideal chip. Our ED^2 is also 42% better than a conservative chip and only 3% worse than the ideal chip.

The remainder of this paper is organized as follows. Section 2 introduces the modeling of process variations in a true 3D memory architecture. Section 3 describes our proposed cache migration scheme to overcome process variations. Section 4 discusses the relate works. Finally Section 5 concludes this paper.

2. ARCHITECTURE AND PROCESS VARIATIONS MODELING IN 3D MEMORY-STACKED MULTICORE CHIP

To understand the impact of process variations (PV) on the performance of a 3D chip stacked with memory, we will first introduce how a PV model is built for a true 3D DRAM. In this section, we will explain the DRAM-on-processor organization, followed by its PV modeling and measurement.

2.1 3D Architecture of DRAM-on-Processor

There are a number of architectural options for a 3D chip that integrates cores and cache (or memory) banks. The first one is to interleave cores and cache banks both in each layer and across layers, forming a staggered layout [3, 19]. This organization avoids direct stacking of active cores through separating them with cooler cache banks. The second design revamps the original 2D circuits into true 3D circuits such that a core or a cache bank spans across all layers of the chip [4,31]. This design further reduces the wire latency within every component of a processor to improve its performance. The last architecture places all cores in one layer that is closest to the heat sink, and all cache banks in the remaining layers [17, 23–26, 39]. This topology has the best capability in heat dissipation, and good scalability in number of layers. In addition, previous study has shown that stacking DRAMs on top of the core layer can result in lowest energy-delay product [39]. Our 3D DRAM-on-processor architecture will also be based on this design.

Core Layer. On the core layer, we incorporate 8 multi-threaded 45nm cores based on the area parameter of Sun UltraSPARC-T2 core [45]. We define the core area at early stage of modeling since it is a constraint to the die area of all the components, such as the memory banks, that will be vertically aligned atop. In UltraSPARC-T2, each core has an area of 12mm^2 in 65nm technology. We scaled this core area to 5.8mm^2 in 45nm technology ($\sim$31% reduction in each dimension). We also assume there is a private 256KB 8-way L2 cache per core. The area of each L2 is $\sim$2.2mm^2, as estimated by CACTI-D [39]. The layout of cores and L2's on the core layer is shown on the lower right corner of Figure 1. As we can see, each core slice (core+L2) occupies about 8mm^2, resulting a total die area of 64mm^2.

Memory Stack. For the memory stack organization, we adopt a design similar to the FaStack 3D technology from Tezzaron [46,47] because it provides faster access speed than direct stacking of conventional 2D memories [17,23,25]. In a conventional 2D memory, every cell array is equipped with a row decoder, sense amplifiers, a row buffer, and column select logic. Those are *peripheral logic* of the memory, and are placed on the same die as the cell arrays. A true 3D memory separates the peripheral logic from the cell arrays. A flat memory bank can be divided into several subbanks which are stacked together to form a 3D bank. Hence, the interconnection wires within a bank are all shorter, which reduces the access latency of the memory. In addition, all peripheral logic are placed in a separate layer which connects with the subbank layers through TSVs, forming a memory of $N+1$ layers where N is number of layers for cell arrays and 1 is the peripheral control logic layer. Such an organization is depicted in Figure 1, where the memory has 4 layers of data banks and 1 layer of peripheral logic, denoted as "Peri", and some other components explained later. The advantage of separating peripherals from the cell banks is that the peripherals can be implemented using fast process technology — the cell layers can be optimized for density and the peripheral circuit can be optimized for speed.

Such a memory architecture is drastically different from conventional off-chip DRAMs. 3D stacking greatly reduces global wiring such as the data transport interconnect. We also placed all peripheral logics on the interface layer using fast process technology. The delay on the peripheral logic including decoder, I/O latch, and I/O driver, are all greatly

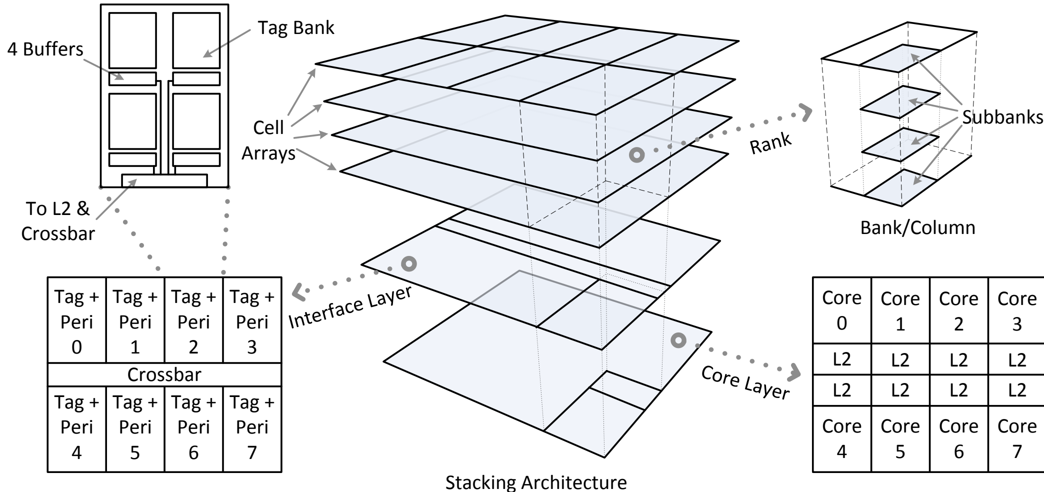


Figure 1: Architecture of a 5-layer 3D memory stacked on top of an 8-core processor.

reduced. Even the wordline driving time is greatly reduced, compared to off-chip DRAMs [27], because the drivers are faster. The bitline discharge time is not reduced proportionally because commodity DRAM cells are used for high density. Therefore, it becomes a more significant portion in the entire memory access time. In our experiment, the bitline with sense amplifier latency amounts to 62% of the total memory bank access time, slightly higher than a 2D on-chip DRAM macro in [27].

Interface Layer. For the 8mm^2 core slice area estimated before, it is difficult to fit in gigabyte of DRAM in the entire column of memory banks atop the core slice. Considering that each core is multithreaded, we choose to use the DRAM banks as the last level cache (LLC) rather than the main memory, similar to the design choice used in [39]. Hence, we need additional transistor budget for the LLC tags which occupy significant die area since the LLC is quite large. It is not beneficial to place these tags in the core layer, as it will expand the die area and increase the cross chip communication delay. Hence, a reasonable design choice is to place them along with the memory peripheral circuits since fast process technology is used here. We term this layer the "interface layer", as shown in Figure 1. Analogously, we also decided to place the interconnection network for this 8-core 3D CMP in this interface layer to keep the chip's footprint small.

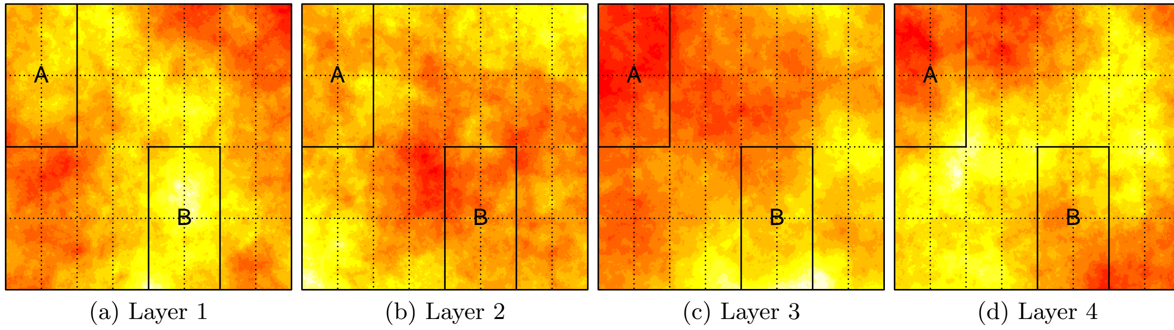
Putting Everything Together. In brief, large memory capacity will result in large tag areas which is constrained by the available area in the interface layer. After testing the tradeoffs between the memory capacity and tag area, we selected a memory rank size of 96MB (1/8th of the total LLC capacity). Each layer within a rank is divided into 4 6MB subbanks for faster access time (vs. 24MB bank). But further dividing the subbanks will increase the peripheral logic area significantly in the interface layer. Unlike traditional caches, we do not use SRAM-based tag implementation as it is too large for our die area. Instead, we use logic-process DRAM (LP-DRAM) for smaller area with good speed. With the LLC cache organization we used, the LP-DRAM tag for the 96MB cache memory occupies about 3.5mm^2 per core in the interface layer.

Since there are 4 layers in each memory rank, and each layer has 4 subbanks, we need to implement 16 sets of peripheral logic for each core in the interface layer. Using CACTI-D, we estimated that these logic and the interconnection in total occupies about 2.67mm^2. The organization of the peripheral logic is shown in the upper left corner of Figure 1: the 4 sets of peripheral logic of one stack of subbanks are placed together, close to the corresponding tag array. We use a dedicated 128-bit bus to connect them with the interface to the crossbar. Note that this interface is also vertically connected to the L2 cache bank in the core layer. The 8-port crossbar in 45nm technology is less than 0.3mm^2. Since the tag plus the peripheral circuits take up about 6.17mm^2 per core, there is sufficient room for 4 128-bit buses (i.e. 512 bits in total), and other necessary control circuits in this layer.

2.2 Process Variations in the 3D DRAM Stack

Within each die, PV can be categorized into *systematic* and *random* variations. Systematic variations are mostly introduced by lithographic aberrations, which has major impact on the effective gate length, L_{eff}. Also, such variations present strong *spatial correlations*, meaning that the variations between two devices that are close to each other are smaller than those that are far apart. Random variations are unrepeatable. They are caused by random doping fluctuation, which has impact on the threshold voltage, V_{th}. Both L_{eff} and V_{th} have critical role in circuit performance. In our PV modeling, we will focus only on the spatially correlated L_{eff} variations and use the results in HSPICE simulation to determine DRAM access latencies. This is because: 1) circuit performance variations are dominated by spatial L_{eff} variations in both normal and low supply voltage region [6]; 2) the relation between L_{eff} and V_{th} variations can be characterized analytically [6, 15, 34], which is inherent in HSPICE.

Modeling Systematic Gate Length Variations. We used VARIUS [34], a PV modeling infrastructure based on the statistic tool `R` [49] and its package `geoR` [32] to model both WID and D2D variations. This model adopts a multivariate Normal (Gaussian) distribution with the well-studied spherical structure [10] for spatial correlations. For a variable, e.g., L_{eff}, that follows the Normal distribution with parameter, mean (μ), variance (σ^2), and density ϕ, VARIUS outputs its variation map. The intuition of μ and σ of WID



(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Figure 3: Sample L_{eff} distribution maps for 4 DRAM cell layers of one chip.

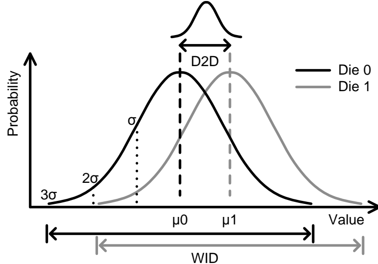


Figure 2: Illustration of WID and D2D variations and their parameters.

and D2D variations can be seen from Figure 2. The density ϕ is a parameter that determines the range of WID spatial correlation. It is expressed as a fraction of a chip's length in one dimension in VARIUS. As the spatial correlation of two devices decreases as their distance grows, ϕ is the distance at which the correlation drops to zero. For example, $\phi = 0.5$ means that the correlation range equals to half of the chip's side length. Hence one device is correlated to all the device within that radius.

The values of μ, σ and ϕ are determined as follows. Since DRAM cells are minimized for device density, we let μ be the minimum gate length for a given technology size. For σ, previous studies [5, 6, 15, 21, 34, 38] used σ/μ ratio to determine σ. This ratio ranges from 3.2% [15] to 10% [6] for different technology nodes. We choose our σ/μ to be 7% for WID L_{eff} variation, similar to that in [21]. For D2D variations, the difference between the μ of each die (e.g., $\mu0$ and $\mu1$ in Figure 2) also follows the Normal distribution. This distribution has the same mean as the nominal minimum L_{eff} we use, but a different variance, σ'. There is little public domain information for σ'. A 5% and 7% for σ'/μ were reported in [21] and [5] respectively. We conservatively take a 2% for σ'/μ in the D2D variation model. The value of ϕ is related to the die size. A large die tends to have relatively small ϕ. For example $\phi = 0.5$ has been used for a 616mm^2 die [12] and a 340mm^2 die [38]. Since our die size is relatively small, we choose $\phi = 1$.

We repeatedly ran VARIUS to generate a large number of maps. Each map corresponds to the L_{eff} distribution of one die. Each map has one million (1000×1000) sample points. Both WID and D2D sample points follow Normal distribution. We then randomly picked 40 maps, and randomly grouped 4 maps together, representing the 4 DRAM layers, to form 10 stacked chips. Figure 3 shows a sample L_{eff} distribution in the 4 DRAM cell layers of a chip. The subbank floorplan of each layer is superimposed on the map. Four neighboring subbanks of the same location in all 4 layers form a memory rank atop a core, as rank A and B shown in the figure. The values of the sample points are converted to different colors on the map. Dark color implies small value, or short L_{eff}, and light color indicates the opposite. We can clearly see from the maps that L_{eff} variations spread across all die, and among different die. Most importantly, when WID and D2D variations are considered together, not only do the subbanks in every layer present L_{eff} variations, but also the subbanks in one vertical rank have a range of L_{eff}. For example, all layers of rank A have relatively short L_{eff}. Therefore, rank A has relatively high average speed. In contrast, rank B has long L_{eff} in Layer 1 and 3. So those two layers are fairly slow, while Layer 2 and 4 are faster. Such a difference gives us motivation to utilize faster banks more frequently for better performance.

Modeling Memory Access Latency Variations. After obtaining the L_{eff} distribution map, we divided each die into 32 subbanks as shown in Figure 3. The latency of each subbank is determined by the slowest cell in this subbank, which corresponds to the longest L_{eff} of all the sample points within the area of each subbank. We then feed those L_{eff} numbers into HSPICE simulation to derive subbank access latency.

We built and simulated the critical path of a DRAM subbank. For the access transistor in a DRAM cell, we used a long-channel, high V_{th} PTM [43,44] nMOS model with thick gate oxide, which are mainly based on parameters in [39] and the CACTI-D source code. The cell capacitor was chosen to be 30fF [28,39], and wordline voltage is boosted to 2.7V [39]. The πRC model was used as the long, very capacitive bitline, and we consider a 40mV bitline voltage swing to be a reasonable sensing margin. In each HSPICE run, the cell transistor was modified with a gate length that represents one subbank, and the resulting latency was then used for that subbank in the subsequent architectural simulations. All HSPICE simulations were performed with a temperature of 90°C.

Modeling Leakage Variations. L_{eff} variations and the induced V_{th} variations also have impact on the leakage of the circuits. When a DRAM subbank is particularly leaky, its refresh period should be more frequent, which affects both performance and energy. To quantify the leakage variations, we also simulated the cell leakage with different L_{eff} values. Due to the inverse proportional relation between L_{eff} and leakage, we found the shortest L_{eff} within each subbank as the most leaky device for HSPICE simulation. Then, we assumed an inverse linear relationship between leakage and refresh period, and scale the refresh period from a normal value of 64ms [39]. For example, if one subbank is two

times leakier than normal, its refresh period will be 64 / 2 = 32ms. In the subsequent architectural simulations, we used subbank level refresh control in a burst refresh mode, meaning that a refresh performs from the first line to the last line continuously. A subbank is marked busy and cannot serve any requests during this time.

Hardware-Awareness of PV. Since PV magnitude can be detected only at the post-fabrication stage, it is necessary for the hardware to detect such variations and store it on-chip for PV-aware configurations. Tezzaron Corporation incorporates a continuous on-chip testing technology which tests all tiny circuits of the entire memory chip at each power-up [48]. Similar technique such as a built-in self-test circuit that can test and record the retention time of each individual cache line after fabrication was also used in previous study [21]. We therefore assume that the stacked chip can be fully tested after fabrication and the tested results can be stored on-chip for PV-tolerant configurations such as one we propose in this paper.

We have discussed how we modeled PV in the DRAM stack of the 3D chip we study. We remark that similar PV also exists in the core layer and the interface layer. Our goal in this paper is to reduce the average access time in the LLC. Therefore, it is additive with the optimizations at the core level. The peripheral layer uses high performance devices, and therefore account for a relatively small part in the entire LLC access time. Therefore, we focus on the latency variations, and the consequent energy results of the DRAM cell layers in this paper, and leave the other factors as future work.

3. NON-UNIFORM CACHE MANAGEMENT

In this section, we first present the performance analysis for the 3D-stacked 5-layer DRAM-on-processor. Next, we introduce our proposed cache management schemes pertaining to the 3D architecture. Before discussing any quantitative results, we first describe our evaluation environment and necessary settings.

3.1 Evaluation Settings

We used Virtutech Simics [50] to model the performance of our 8-core 3D DRAM stacked processor. Major parameters are detailed in Table 1. We extended the g-cache module in Simics with PV features such as different bank access times. In addition, as we will explain later, we modeled the contention in cache subbank accesses as well as the crossbar interconnect for our PV-tolerant designs. Both L1 and L2 are private, and LLC is shared. We used snoopy MESI protocol in L2 to maintain the cache coherence. For the private L2, a miss would trigger a snoop request to all L2s, and then waits for the response back. Such a coherence transaction can take a long time. Fortunately, our LLC has large capacity, and it is highly likely that the local L2 miss can be satisfied in the LLC. Therefore, we parallelize these two operations for performance benefit.

As we can see from Table 1, the nominal LLC subbank latency is 15 cycles. However, the speed of a subbank is determined by the slowest device in the subbank. Hence, the subbank speed distribution has a larger mean than 15 cycles. We ran simulations on 10 modeled chips, randomly generated using the methodology described before. Figure 4 shows the histogram of the subbank latency as the result of PV, collected from all four memory layers of all 10 chips we modeled. The results show that a majority of the subbanks have latencies longer than 15 cycles, creating challenges to PV-tolerant designs. The last subtlety is the latency model for the interface layer. We charge 3 cycles for source-to-destination traffic through the crossbar in an uncontended network. Contention latencies will be further added during the simulation. For tag, data bank, and row buffers that are immediately next to the crossbar interface (refer to Figure 1), there is only 1 cycle delay on the wire. For others, there are 2 cycle delays. These values are important source of overhead in our proposed migration schemes.

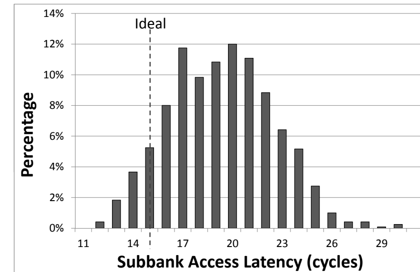


Figure 4: DRAM subbank latency distribution as the result of PV.

Benchmarks. We selected 5 multi-threaded workloads from PARSEC [51] and 10 workloads from SPECCPU-2006. Those workloads are memory intensive, and therefore, more sensitive to process variations in LLC. All workloads were compiled with gcc 4.1.0. For PARSEC workloads, we used `simlarg` as the input and simulated the code region of interest from the beginning to the end. For SPECCPU-2006 workloads, we skipped the initialization phase, and simulated 2 billion instructions afterwards. The details of the workloads are listed in Table 2. For multi-threaded workloads, we applied *page coloring* [8] technique to enhance data locality for performance benefit. We ran the workloads twice, the first time distinguishing private and shared pages and the second time pinning private pages to their local LLC banks. For single-threaded workloads, all data are located in the memory rank atop the core.

3.2 Static PV-Tolerant LLC

We use two baselines as our basis for comparison. The first one is an ideal chip that has no PV, i.e., a chip with uniform LLC. This baseline gives us the ideal performance that we would like to achieve in presence of PV. The second baseline considers the impact of PV, and uses the slowest subbank's latency as the nominal LLC bank access latency. This baseline obviously produces the worst performance. Figure 5 compares the performance of the two baselines, normalized to the ideal one. The error lines show the performance range of the 10 chips we modeled, and the bars show their average performance. In the "l3-worst" results, we did not use the real slowest bank in the entire LLC as it is typically too slow to make the comparison fair. In reality, cache or memories are typically built with redundancy, and faulty memory lines can be turned off and remapped to the redundant memory lines. We also assume there is such redundancy for PV purposes so that the slowest subbanks are disabled. Hence, in our measurements, we first eliminated the 4 slowest cache lines in each subbank, then used the third slowest subbank's speed to produce the results. As we can see that

CPU	8 cores, 3GHz, 2-issue, in-order
Private L1 I/D	32KB/core, 8-way, 64B line, 2-cycle hit time
Private L2	256KB/core, 8-way, 64B line, 2-cycle tag lookup, 3-cycle data hit time
Shared LLC Organization	distributed 3D stacking, 4 data cell layers, 1 interface (peripheral) layer tag array: 4 subbanks per core data array: 4 subbanks per core per layer 6MB tag array on the interface layer, 32-bit per tag entry
LLC Configurations	768MB, 48-way, 3-way per subbank 512B line, 5-cycle tag lookup, 15-cycle data hit time 2-cycle delay for tag and data subbank not immediate to the crossbar
Interconnect	8-port crossbar, 16B wide, bidirectional, 3-cycle end-to-end uncontended latency
Main Memory	16GB, 2 channels, 160-cycle for the critical block

Table 1: Baseline chip configurations without PV.

SPEC2006	astar	bwaves	gcc	GemsFDTD	lbm	leslie3d	libquantum	mcf	soplex	sphinx3
memory	46M	820M	45M	838M	407M	81M	36M	29M	92M	22M
L2 MPKI	5.4	10.2	6.8	18.3	23.2	16.8	33.8	69.3	24.6	12.7

PARSEC	canneal	facesim	ferret	streamcluster	x264
memory	162M	210M	71M	16M	90M
L2 MPKI	21.0	4.8	3.1	8.7	2.2
instructions	1700M	28500M	38000M	35000M	24300M

Table 2: Simulated workloads. "memory" stands for memory footprint. "L2 MPKI" stands for L2 misses per kilo-instructions. "instructions" stands for the number of instructions simulated. All SPEC2006 workloads are run for 2 billion instructions.

the pessimistic baseline is 6∼29%, with an average of 16% worse than the ideal baseline. The performance degradation is seen more for the single-threaded workloads because their L2 MPKIs are higher which indicates that their LLC demands are higher.

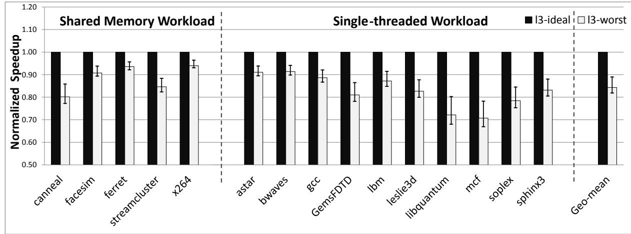


Figure 5: Performance of the pessimistic baseline normalized to the ideal baseline.

With PV-tolerant LLC designs, we expect that the performance will win over the pessimistic baseline, but inferior to the ideal baseline. Hence, our objective is to approach the performance of the ideal baseline. The first PV-tolerant design is the straight forward static NUCA design, much like that in a 2D CMP. That is, every subbank has its own speed, irrespective of the slowest subbank speed. Therefore, some subbanks will indeed be faster than the nominal speed. However, one design complication is that the cache requests may return out-of-order since each subbank has different access latency. This may create contention in the interconnection channel shared among multiple subbanks. In our design, this is the dedicated bus that connects the 4 row buffers of each column of subbanks to the TSV/crossbar port in the interface layer (refer to Figure 1). The contention comes from the vertically aligned 4 memory subbanks. We added a local arbiter to arbitrate their requests for this bus. Figure 6 shows the performances of this scheme compared to the two baselines. As we can see, all workloads immediately benefit from a PV-tolerant design. The average slowdown compared to the ideal baseline is now 6%.

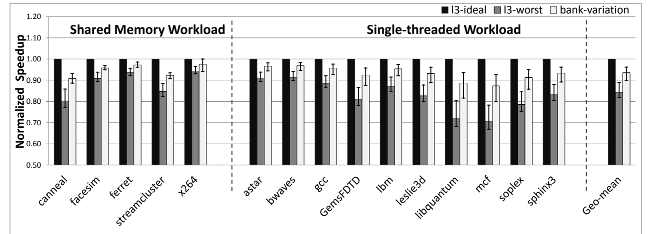


Figure 6: Performance of the static PV-tolerant LLC, normalized to the ideal baseline.

3.3 Dynamic PV-Tolerant LLC

In static PV-tolerant design, data are statically mapped to a subbank. The data in a slow subbank will always be accessed with long latency. We can improve this by applying data migration, similar to the DNUCA design in a 2D CMP [16]. The idea is to migrate data from slow subbanks to fast subbanks. For example, if an LLC access hits in a slow subbank, after the data is sent back to the core, it is moved immediately to a fast subbank such that next time the data can be fetched with lower latency. Note that the tags are moved as well to ensure correct indexing. The question is where to migrate the data, as there might be many subbanks that are faster.

Bank Latency Based Migration Policy. An intuitive migration scheme is to always move the data to the fastest subbank. That is, all 16 subbanks in a memory rank are sorted by their access latencies in ascending order from Bank-0 to Bank-15. Once a cache line is accessed in a slow subbank, it is migrated to the fastest subbank (Bank-0). The victim cache line from Bank-i are evicted to Bank-(i+1), $0 \leq i \leq 15$, as illustrated in Figure 7(a). However, this scheme in fact degrades the performance on average, as can be seen from the data series labeled "bank-migration" in Figure 8. The results show that it is even 7% worse than the static PV-tolerant scheme. This is because a lot of contention was created when all rest 15 subbanks move their

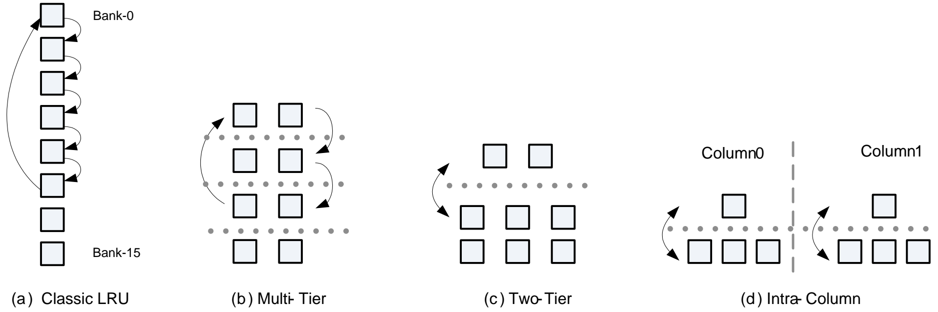


Figure 7: Illustrations of different dynamic data migration policies.

data into Bank-0. All migration activities compete for the free time of Bank-0, generating long wait time due to this contention. What makes things worse is that Bank-0 is now also highly demanded as all data are supposed to be located there. Hence, every LLC access is first directed to Bank-0 then other subbanks, generating even more traffic to Bank-0.

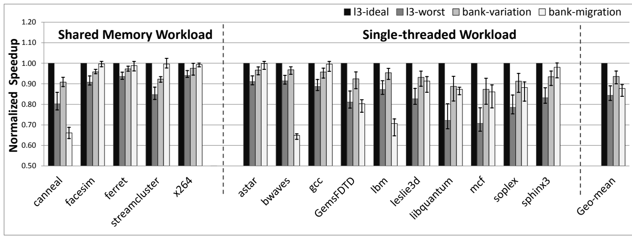
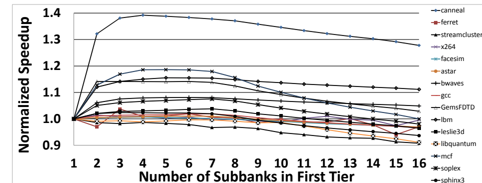


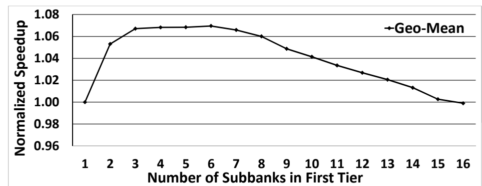
Figure 8: Performance of the latency-based migration policy, normalized to the ideal baseline.

Tiered Migration Policy. If we take another look at Figure 8, we find that some workloads such as `facesim`, `ferret`, `gcc`, `sphinx3` etc. did see improvements with the above migration policy. Some of them are already as good as the ideal baseline. Those are the workloads that have less than ∼10 L2 MPKI, as indicated in Table 2, meaning that their demand on LLC is relatively low. This observation motivates us to revise the migration scheme to offload the requests from Bank-0 to maybe the second, third, etc. fastest subbanks. More formally, we divide the subbanks in one rank of memory into several tiers according to their speed, and each tier has several subbanks of similar speed. Migration is done such that data is moved from a slow tier to the fastest tier, and evictions go in the opposite direction. Within each tier, cache lines are installed in a Round-Robin manner among different banks. Therefore, the pressure originally on Bank-0 is now evenly distributed into several subbanks, greatly reducing the likelihood of subbank contention. This is illustrated in Figure 7(b).

Next, we study how to divide the subbanks into tiers. We evaluate how many tiers are necessary, and what size of each tier should be. We first start from a simple 2-tier design: a fast tier and a slow tier. We vary the size of the fast tier from 1 to 16, and leave the remaining subbanks to the slow tier. For example, the latency based migration is simply a 2-tier scheme where the fast tier has only Bank-0, and all the rest 15 subbanks are in the slow tier. We measured the performance of all partitions and normalized them to this configuration. Results are shown in Figure 9(a). As we can see, there is a tradeoff between the tier size and performance, as the initial size increase in the fast tier does help to improve the performance for most benchmarks due to load distribution. However, when the fast tier size is larger than the slow tier, performance starts to decrease. This is because when the fast tier is bigger, the average LLC access time is also larger which in turn harm the performance. There are also a few workloads, such as `sphinx3` that have nearly monotone decreasing performance. Those workloads generally have small LLC demand and less contention in Bank-0. Therefore, the smaller the fast tier, the lower the LLC average access time, and the higher the performance. Figure 9(b) summarizes Figure 9(a) using Geometric means of all workloads. This curve clearly shows the best choices for the fast tier: 3∼6 fastest subbanks give the highest performance. Moreover, we also tested the potential of having a dynamic varying fast tier assuming that the best tier partition can be found dynamically. The Geometric mean of this dynamic optimal value is well below 1% away from the static partition of 3∼6 subbanks. This implies that it is not necessary to perform dynamic searching for the best partition at runtime since static partition is sufficiently good.



(a)



(b)

Figure 9: Performance variations (a) and summary (b) for a 2-tier migration scheme.

Finally, we keep increasing the number of tiers to see if further performance improvements can be obtained. Figure 10 compares the performance for 2-tier, 3-tier and 4-tier migration schemes. As we can see, there is no clear performance benefit in increasing the tier count. Two-tier enabled migration scheme is adequate and the migration operation is also simple. A cache line swap between the two tiers is enough

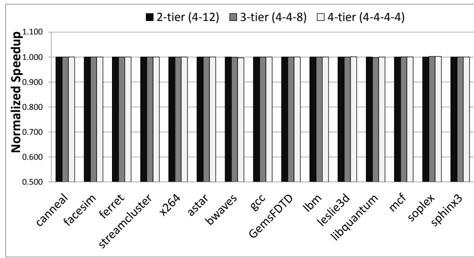


Figure 10: Sensitivity study of tier numbers.

to gain performance benefit. Our two-tier partition has 4 subbanks in the fast tier, which amount to 6MB×4×8ranks =192MB memory space. This is fairly large to hold most of our workload's working sets. For workloads that are even larger, it may be advantageous to have more tiers. In all the studies next, we use a 2-tier (4-12) partitioned configuration.

The 2-tier migration scheme we developed achieves significant performance improvements, as shown in Figure 11. For all workloads except `libquantum`, the maximum performances of the 10 test chips we generated surpass the ideal baseline. The highest is seen for `mcf` which is 7% faster than the ideal baseline. This is because the 2-tier migration scheme can fully utilize the fast subbanks such that the effective LLC access latency is even shorter than the ideal baseline. For the average performance of each workload (the height of the bars), there are also 3 workloads: `GemsFDTD`, `bwaves`, `mcf`, and `sphinx3` that are faster than the ideal baseline. The Geometric mean of all workloads achieves 99.55% of the performance of the ideal baseline. Having a fast tier of 4 subbanks are particularly effective for workloads that have high LLC requirement (or high L2 MPKI) such as `canneal`, `GemsFDTD`, `lbm` and `mcf` as their performances boost dramatically from "bank-migration" to "bank-migration-2-tier" in the figure.

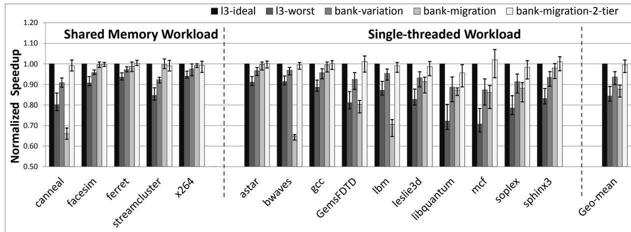


Figure 11: Performance of the proposed 2-tier migration technique, normalized to the ideal baseline.

Energy Conservation: Intra-Column Two-Tier Migration. Having seen the great performance advantages of the 2-tier migration scheme, we now turn into its energy consumption as the migration activities are extra energy overhead. First, let us understand how much migration activities are introduced to the memory. This is reported as the percentage of LLC accesses that trigger migrations in Figure 12(a). Since our fast tier has 4 subbanks (192MB for entire LLC), most workload's working set can be fit into this capacity. Hence, many workloads do not have noticeable migration overhead — they are only for the cold start stage. For the 6 workloads that have >1% migration activity, we further study their LLC regular dynamic energy consumption vs. migration energy. The results are in Figure 12(b). The data series labeled with "intra-column" will

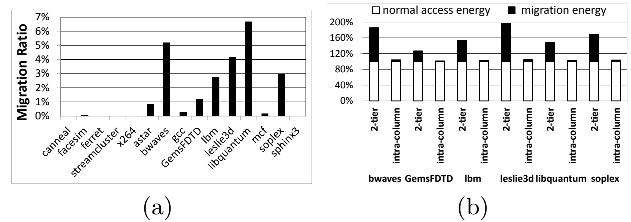


Figure 12: (a) Percentage of migration activities. (b) Energy comparison for 2-tier migration and intra-column migration.

be discussed later. We can see that for 2-tier design, the migration energy can be more than 80% of the regular access energy for `leslie3d`. Hence, it is necessary to revise our 2-tier migration scheme to lower this part.

To reduce the migration energy, we have observed that a significant portion of the migration energy is consumed on the wires in the interface layer connecting the four row buffers and tag arrays (refer to Figure 1). Table 3 lists the energy breakdown for a LLC access. As we can see, the 0.325nJ, which is the wires in the interface layer is the largest amount in the table. Since 3D memory design has shorter interconnection wires due to stacking, any 2D wires that cannot be eliminated become more pronounced. The energy spent on the wires in the interface layer is due to the movement of data between subbanks in different columns. Hence, we revise our 2-tier migration scheme into an *intra-column* migration scheme. Instead of migrating among different columns via the wires on the interface layer, we enforce the migration to happen within each column. There are still 2 tiers in the memory rank. The difference now is that the fast tier is composed of the fastest subbank *in each column*. A hit into a slow subbank will only move the data to the fastest subbank in its own column. Hence, all migrations happen only within each column of subbanks. This is depicted in Figure 7(d).

L3 Energy Component	Energy (nJ)
Data Subbank Read	0.065
Data Subbank Write	0.087
Subbank Wire (64B, one-way)	0.325
Tag Read	0.052
Row Buffer Swap	0.230

Table 3: L3 Energy consumption component breakdown. Obtained using CACTI-D.

The intra-column migration technique saves significant energy, compared to the original 2-tier migration scheme. The comparisons are shown in Figure 12(b). As we can see, intra-column migration has nearly negligible migration energy overhead, which is more than ∼20 times lower than the 2-tier design for all workloads. What is more important is that the intra-column migration, though it changed the subbanks in the fast tier, was not hurt in performance. This is shown in Figure 13. Both 2-tier migration schemes are very close to each other for all workloads. The intra-column migration has a slightly higher variance in speedup than the 2-tier migration. The average of intra-column is 98.93% of the ideal baseline. They are better than the pessimistic baseline by 18%. Finally, we also measured the energy-delay product for the entire chip using methodologies similar to [39] and compared them among different PV-

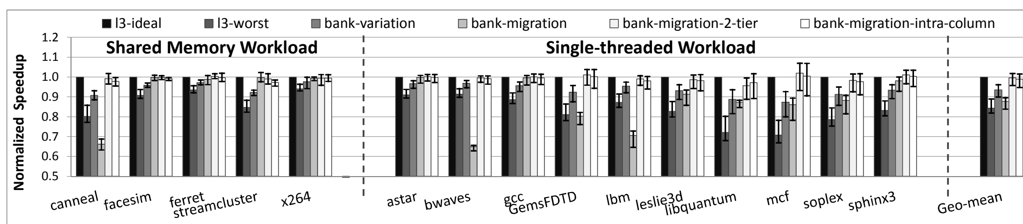


Figure 13: Performance of the improved intra-column 2-tier migration technique, normalized to the ideal baseline.

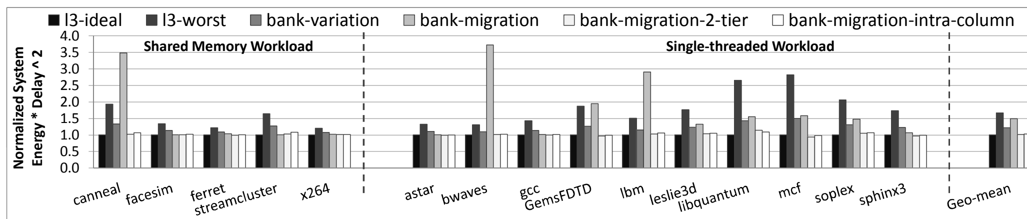


Figure 14: Energy-Delay product results.

tolerant schemes. The results are shown in Figure 14. Since our proposed intra-column 2-tier migration scheme achieves almost identical performance as the ideal baseline, and its energy overhead is also negligible, it follows naturally that the energy-delay^2 product is only 3% away from the ideal baseline. They are however 42% better than the pessimistic baseline. We therefore conclude that this is the best PV-tolerant cache management scheme.

4. PRIOR ART

The concept of Non-uniform Cache Architecture (NUCA) was proposed by Kim *et al.* [16] to handle the increasing global wire delay. They study different cache line migration policies, cache line mapping and tag searching techniques for a single-core architecture. Chishti *et al.* [7] proposed NuRapid, a NUCA design with decoupled tag array and data array. Cache line can be relocated to a different data array bank with an indirect pointer from the tage entry. In multi-core architectures, Cho *et al.* [8] proposed a software approach to place data close to its core. With a special cache mapping scheme, cache line placement can be managed through page coloring at the OS level.

There are also a number of recent works that are related to managing the non-uniformity of cache accesses in 3D chips [19], including those that utilize emerging memory technologies such as Phase Change Memories and Magnetic Memories [37, 41]. However, the fundamental cache non-uniformity of those works are still two-dimensional such that they are managed within one layer. Migrating data vertically did not benefit because vertical communication latency was uniform (well below $1ns$) due to the small die thickness. However, as we can see from our analyses, such vertical data migration is critical to performance with process variations.

5. CONCLUSION

We modeled PV in a 3D DRAM-stacked multicore chip. With PV, the speed of DRAM subbanks has a wide range of values due to both WID and D2D variations. Hence, it is important to consider the impact of PV on the performance in a 3D chip with DRAM stacks. Our evaluation indicates that the performance of a 3D chip with PV can be degraded by 16% if the speed of one of the slowest subbank is used as the nominal subbank speed. We proposed a cache management scheme based on data migration between tiers within a column of subbanks. Our scheme can overcome PV in memories such that the resulting speed is only 0.45% away from a chip with no PV. The energy overhead is also minimized.

6. REFERENCES

[1] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, K. Roy, "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies," *IEEE Transactions on Very Large Scale Integrated Systems,* vol. 13, pp. 27-38, 2005.

[2] A. Agarwal, B. C. Paul, S. Mukhopadhyay, K. Roy, "Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture," *IEEE Journal of Solid-State Circuits,* 40(9), pp. 1804-1814, 2005.

[3] M. Agasthi, V. Venkatesan, R. Balasubramonian, "Understanding the Impact of 3D Stacked Layouts on ILP," *Journal of Instruction-Level Parallelism,* Vol. 9, pp. 1-27, 2007.

[4] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCaule, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, C. Webb, "Die Stacking (3D) Microarchitecture," *International Symposium on Microarchitecture,* pp. 469-479, 2006

[5] K. A. Bowman, S. G. Duvall, J. D. Meindl, "Impact of Die-to-Die And Within-die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration," *IEEE Jounal of Solid-State Circuits,* Vol. 37, No. 2, pp. 183-190, 2002.

[6] Y. Cao, L. T. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach," *Design Automation Conference,* pp. 658-663, 2005.

[7] Z. Chishti, M. Powell, T. N. Vijaykumar, "Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures," *International Symposium on Microarchitecture,* 2003

[8] S. Cho, L. Jin, "Managing Distributed, Shared L2 Caches Through OS-Level Page Allocation," *International Symposium on Microarchitecture,* pp. 455-465, 2006.

[9] E. Chun, Z. Chishti, T. N. Vijaykumar, "Shapeshifter: Dynamically Changing Pipeline Width and Speed to Address Process Variations," *International Symposium on Microarchitecture,* pp. 411-422, 2008.

[10] N. Cressie, "Statistics for Spatial Data", *Wiley,* 1993.
[11] A. Das, B. Ozisikyilmaz, S. Zademir, G. Memik, J. Zambreno, A. Choudhary, "Evaluating the Effects of Cache Redundancy on Profit," *International Symposium on Microarchitecture,* pp. 388-398, 2008.
[12] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, C. Spanos, "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization," *International Symposium on Quality Electronic Design,* 2005.
[13] X. Fu, T. Li, J. Fortes, "NBTI Tolerant Microarchitecture Design in the Presence of Process Variation," *International Symposium on Microarchitecture,* pp. 398-410, 2008.
[14] X. Fu, T. Li, J. Fortes, "Soft Error Vulnerability Aware Process Variation Mitigation," *High-Performance Computer Architecture,* pp. 2009.
[15] S. Hebert, D. Marculescu, "Variation-Aware Dynamic Voltage/Frequency Scaling," *High-Performance Computer Architecture,* pp. 2009.
[16] C. Kim, D. Burger, S. W. Keckler, "An Adaptive, Non-uniform Cache Structure for Wire-Delay Dominated On-Chip Caches," *International Conference on Architectural Support for Programming Languages and Operating Systems,* pp. 211-222, 2002.
[17] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, K. Flautner, "PicoServer: Using 3D Stacking Technology to Enable a Compact Energy Efficient Chip Multiprocessor," *International Conference on Architectural Support for Programming Languages and Operating Systems,* pp. 117-128, 2006.
[18] J. P. Kulkarni, K. Kim, S. P. Park, K. Roy, "Process Variation Tolerant SRAM Array for Ultra Low Voltage Applications," *Design Automation Conference,* pp. 108-113, 2008.
[19] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, M. Kandemir, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," *International Symposium on Computer Architecture,* pp. 130-141, 2006.
[20] X. Liang and D. Brooks, "Mitigating the Impact of Process Variations on CPU Register File and Execution Units," *International Symposium on Microarchitecture,* pp. 504-514, 2006.
[21] X. Liang, R. Canal, G.-Y. Wei, D. Brooks, "Process Variation Tolerant 3T1D-Based Cache Architectures„" *International Symposium on Microarchitecture,* pp. 15-26, 2007.
[22] X. Liang, G.-Y. Wei, D. Brooks, "ReVIVaL: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," *International Symposium on Computer Architecture,* pp. 191-202, 2008.
[23] C.C. Liu, I. Ganusov, M. Burtscher, S. Tiwari, "Bridging the Processor-Memory Performance Gap with 3D IC Technology," *IEEE Design and Test of Computers,* 22(6), pp. 556-564, 2005.
[24] G. Loh, "3D-Stacked Memory Architecture for Multi-Core Processors," *International Symposium on Computer Architecture,* pp. 453-464, 2008.
[25] G. L. Loi, B. Agarwal, N. Srivastava, S. Lin, T. Sherwood, "A Thermally-Aware Performance Analysis of Vertically Integrated (3D) Processor-Memory Hierarchy," *Design Automation Conference,* pp. 991-996, 2006.
[26] N. Madan, L. Zhao, N. Muralimanohar, A. Udipi, R. Balasubramonian, R. Iyer, S. Makineni, D. Newell, "Optimizing communication and capacity in a 3D stacked reconfigurable cache hierarchy," *International Symposium on High Performance Computer Architecture,* pp. 262-274, 2009.
[27] R. E. Maatick, S. E. Schuster, "Logic-based eDRAM: origins and rationale for use," *IBM Journal of Research and Development,* pp. 145-165, 2005.
[28] W.Mueller, et al., "Challenges for the DRAM Cell Scaling to 40nm" *IEEE International Electron Devices Meeting,* 4 pages, Dec 2005
[29] S. R. Nassif, "Modeling and Forecasting of Manufacturing Variations," *Asia and South Pacific Design Automation Conference,* pp. 145-149, 2001.
[30] S. Ozdemir, D. Sinha, G. Memik, J. Adams, H. Zhou, "Yield-Aware Cache Architectures," *International Symposium on Microarchitecture,* pp. 15-25, 2006.
[31] K. Puttaswamy, G. H. Loh, "Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance 3D Integrated Processors," *International Symposium on High Performance Computer Architecture,* pp. 193-204, 2007.
[32] P. Ribeiro Jr., P. Diggle, "geoR: A Package for Geostatistical Analysis," *R-NEWS,* vol. 1, no. 2, 2001.
[33] S. Sarangi, B. Greskamp, A. Tiwari, J. Torrellas, "EVAL: Utilizing Processors with Variation-Induced Timing Errors," *International Symposium on Microarchitecture,* pp. 423-434, 2008.
[34] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, J. Torrellas, "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," *IEEE Transactions on Semiconductor Manufacturing,* Vol. 21, No. 1, 2008.
[35] J. Singh, J. Mathew, D.K. Pradhan, S.P. Mohanty, "Failure Analysis for Ultra Low Power Nano-CMOS SRAM Under Process Variations," *International SOC Conference,* pp. 251-254, 2008
[36] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical Analysis and Optimization for VLSI: Timing and Power," New York Springer, 2005
[37] G. Sun, X. Dong, Y. Xie, J. Li, Y. Chen, "A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs," *International Symposium on High Performance Computer Architecture,* pp. 239-249, 2009.
[38] R. Teodorescu, J. Torrellas, "Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors," *International Symposium on Computer Architecture,* pp. 363-374, 2008.
[39] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, N. P. Jouppi, "A Comprehensive Memory Modeling Tool and its Application to the Design and Analysis of Future Memory Hierarchies," *International Symposium on Computer Architecture,* pp. 51-62, 2008.
[40] A. Tiwari, S. R. Sarangi, J. Torellas, "ReCycle: Pipeline Adaptation to Tolerate Process Variation," *International Symposium on Computer Architecture,* pp. 323-334, 2007.
[41] X. Wu, Y. Xie, J. Li, L. Zhang, E. Speight, R. Rajamony, "Hybrid Cache Architecture with Disparate Memory Technologies," *International Symposium on Computer Architecture,* 2009.
[42] S. Lee, C. Choi, J. Kong, W. Lee, J. Yoo, "An Efficient Statistical Analysis Methodology and Its Application to High-density DRAMs," *International Conference on Computer-Aided Design,* pp. 678-683, 1997
[43] W. Zhao, Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Early Design Exploration," *IEEE Transactions on Electron Devices,* Vol. 53, No. 11, pp. 2816-2823, 2006.
[44] Arizona State University, "Predictive Technology Model (PTM)," http://www.eas.asu.edu/∼ptm/
[45] UltraSPARC T2 Processor, http://www.sun.com/processors/UltraSPARC-T2/
[46] Tezzaron Semiconductors, FaStack Memory, http://www.tezzaron.com/memory/FaStack_memory.html
[47] Tezzaron Semiconductors, 3D Stacked DRAM, http://www.tezzaron.com/memory/Overview_3D_DRAM.htm
[48] Tezzaron Semiconductors, Bi-STAR Technology, http://www.tezzaron.com/technology/Bi_STAR.htm
[49] R Development Core Team, "R: A Language and Environment for Statistical Computing," *R Foundation for Statistical Computing,* http://www.R-project.org, 2006.
[50] Virtutech Simics, http://www.virtutech.com
[51] The PARSEC Benchmark Suite, http://parsec.cs.princeton.edu