Mitigating the Effects of Process Variations: Architectural Approaches for Improving Batch Performance

Abhishek Das, Serkan Ozdemir, Gokhan Memik, Joseph Zambreno and Alok Choudhary
Electrical Engineering and Computer Science Department,
Northwestern University. Evanston, IL
{ada829, soz463, memik, zambro1, choudhar}@eecs.northwestern.edu

ABSTRACT

As transistor feature sizes continue to shrink into the sub-90nm range and beyond, the effects of process variations on critical path delay have amplified. A common concept to remedy the effects of variation is speed-binning, by which chips from a single batch are rated by a discrete range of frequencies. In this paper, we argue that under these conditions, architectural optimizations should consider their effect on the "batch" of microprocessors rather than aiming at increasing the performance of a single processor. We first show that the critical paths are mostly determined by the level 1 data caches on a set of manufactured microprocessors. Then, we propose three new microarchitectural techniques aimed at masking the effects of process variations on level 1 caches. The first two techniques allow individual high-latency cache lines spanning single or multiple sets to be disabled at the post-manufacture testing stage. The third approach introduces a small substitute cache associated with each cache way to replicate the data elements stored in the high latency lines. Our new schemes can be effectively used to boost up the overall chip yield and also shift the chip binning distribution towards higher frequencies. To make a quantitative comparison between the different schemes, we first define a metric called batch-performance that takes into account the chip yield and frequency of chips in each bin. We then analyze our proposed schemes and show that the resizing schemes and the substitute cache can increase the batch-performance by as much as 5.8% and 11.6%, respectively.

Keywords: Device Variability, Process Variations, Cache Architecture, Fault-tolerance, Computer System Design.

1. INTRODUCTION

With the aggressive scaling of silicon technology, parameter variations are expected to have significant impact on the power, performance, and reliability of microprocessors. As transistors are reduced in size, it becomes harder to control variations in device parameters such as channel length, gate width, oxide thickness, and device threshold voltage. These fluctuations in the process parameter distributions or simply process variations cause increased variability in circuit performance and are likely to be more dominant in sub-90 nm domain. Even in a relatively mature technology like 130 nm, these variations are known to result in as much as 30% decrease in maximum frequency and 500% increase in leakage power [8]. For newer technologies, these variations can be even higher: 20-fold increases in leakage have been reported for 90nm [4]. Process variations consist of With-in-Die (WID) and Inter-Die or Die-to-Die (D2D) parameter variations. As a direct impact of this, a chip may under-perform or turn out to be more leaky than a certain threshold and hence may be eventually dropped resulting in effective yield loss. A common practice to remedy the effects of process variations is *speed-binning* (Figure 1). Speed-binning is usually performed by testing each manufactured chip separately over a range of frequency levels until it fails. As a result of the inherent process variations, the different processors fall into separate speed bins, where they are rated and marketed differently. This process helps the chip manufacturer create a complete product line from a single design.

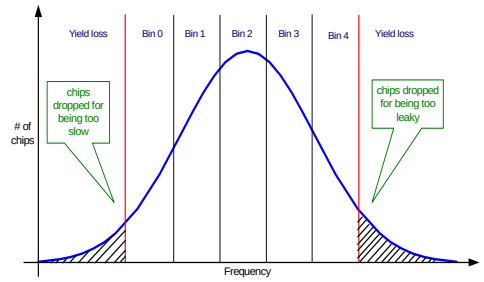

Figure 1. Frequency binning in modern microprocessors.

In contrast to speed-binning, architectural changes made for performance enhancements are generally analyzed by considering its effects on high-level metrics such as instructions-per-cycle (IPC) and/or cycle time. However, because of the effects of process variations, different chips can have different post-fabrication frequencies irrespective of the changes made. Hence IPC and clock cycle are not enough to judge the effects of an architectural modification on the performance of a whole batch of chips. As a result, we need to establish new metrics when the process variations are considered. Chip yield is one obvious metric, as the continuing downward scaling of transistor feature sizes has made fabrication considerably more difficult and expensive [14, 17, 15]. However, an approach that optimizes solely for yield, would not take into account the fact that CPUs concurrently manufactured using a single process are routinely sold at different speed ratings and prices. For example, from a manufacturer's perspective, having a 20% yield where the chips have 2.4 GHz frequency may be more desirable than having a 50% yield where the processors have 1.0 GHz maximum clock frequency. In this paper, we propose a new metric called

batch-performance (BP) to capture the effects of architectural changes. Batch-performance corresponds to the average performance of the batch of chips manufactured using a given architecture, therefore captures the distribution of chips as well as the chip yield. In Section 5.2, we provide a detailed explanation of this metric.

In addition to proposing this new metric, we investigate the effects of cache resizing schemes on batch-performance and then propose a novel scheme called ***Substitute Cache (SC)*** that aim at improving overall binning distribution with post-fabrication modifications. First, we study the impact of two resizing schemes namely, ***One-Way Sizing (OWS)*** and ***Multi-Way Sizing (MWS)***. OWS disables selected cache word lines with critical or near-critical delay. By disabling the high-latency word lines after manufacturing, this approach improves the yield at the low end of the frequency distribution, and also increases the likelihood that any valid chip will be placed in a higher-frequency bin. MWS extends this idea to a word line as it spans multiple sets in the cache, working off the theory that a high-latency word line in a single cache set would also likely be a critical path in the other sets. In addition to these schemes, we also propose the SC scheme, in which the level 1 (L1) cache is augmented with a small *substitute cache* storing the most critical cache words. With the help of minimal control logic, the processor can fetch data from SC instead of the main data array whenever a read/write access is made to these critical words. Hence access latency is minimized with no extra cache misses.

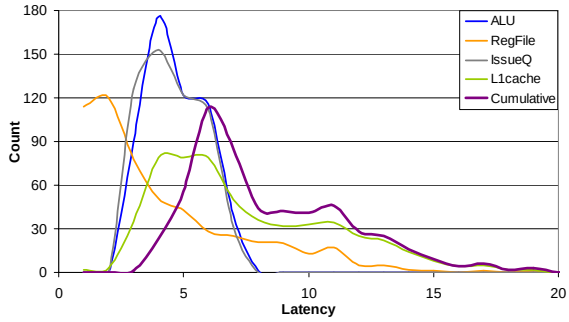

Figure 2. Delay distribution of each microarchitectural unit for a set of 1100 processors. The cumulative distribution gives the critical path of the chip.

The reason why we emphasize on caches is because L1 caches are likely to be the critical path under process variations. Figure 2 illustrates the latency distributions of various architectural units; for a set of 2000 simulated chips (the details of the modeling framework are described in Sections 2 and 3). The analysis reveals that 58.9% of the critical paths lie in the L1 cache. Therefore, in this work, we focus on the level 1 cache.

Although our techniques will have impact on various design stages, such changes remain minimal. The cache resizing schemes may impact the cycle per instruction (CPI) of the processors, however, as we show in Section 4.1, these effects are limited to 0.3% on average across SPEC2000 applications. On the other hand, the proposed schemes have significant impact on the binning distribution. As we describe in Section 5, we simulate the effects of process variations on yield and binning distribution using SPICE. These results are fed into parameterized binning models, which show that applying our OWS approach to current processor architectures could lead to a 4.32% increase in batch-performance. Similarly, MWS shows a 5.83% increase. For the SC technique, this increase can reach 11.59%. These gains are achieved mostly through an increase in the number of chips in the higher-frequency bins. It should be noted that alternative approaches like Error Correcting Codes (ECC) could not be used for our purpose. The errors in timing violations generally cause high number of bit flips, under which realistic ECC schemes fail.

The remainder of this paper is organized as follows. In the following section, we describe our methodology for measuring the effect of process variations on a representative processor architecture. Section 3 describes how we model the binning. In Section 4, we present the proposed cache architectures in detail. Experimental results detailing the effects of our approaches on the binning distribution with its implications on the overall batch performance are presented in Section 5. Related work pertaining to this area is presented in Section 6. Finally, the paper is concluded in Section 7 with a brief summary.

2. PROCESS VARIATIONS AND MODELING

This section presents a description of the cache architecture we use in this paper and describes how we model process variations. Note that at this stage we confine ourselves to analyzing level 1 caches only.

2.1 Processor Model

To model a processor core, we have taken into account the classical 7-stage pipeline in Alpha-21364 (EV7) architecture. The main critical components of our processor are the Issue Queue, the Integer Execution Unit, the Register File, and the L1 Data cache. All the above components were modeled in SPICE using the 45nm BPTM technology models [5]. The issue queue is based on that of EV7 and has 20 entries. The register file is an 80-entry structure with 4 read and 2 write ports. The integer execution unit is modeled using the netlist generated after synthesizing the corresponding component in the Sun OpenSPARC [27]. Our L1 cache is a 32 KB 4-way set associative cache, the model of which is based on the architecture described by Amrutur and Horowitz [3]. Each of the 4 ways of our cache is further divided into 4 banks. Each bank has 128x128 cells or storage bits. Thus, each bank has exactly 128 rows (i.e., lines) and can hold 2-KB of data. The bitline delays are reduced by partitioning them into two. To account for the effects of submicron technologies on circuit behavior, we added coupling capacitances at three places in the cache: between the lines in the address bus from the driver, between parallel wires in

the decoder, and between bit-line and bit-line bar. Furthermore, these lines as well as global and local word lines are replaced by distributed RC ladders representing the local interconnect wires inside the cache.

2.2 Simulating Process Variations

Process variations can be defined as statistical variations in circuit parameters like gate-oxide thickness, channel length, Random Doping Effects (RDE), etc., due to the shrinking process geometries [4, 16]. As mentioned before they mainly consist of D2D and WID variations. D2D variation refers to the variation in process parameters across dies and wafers, whereas WID variation is the variation in device features within a single die, causing non-uniform characteristics inside a chip. Independent of their types, process variations generally fall into two categories: spatially-correlated variations where devices close to each other have a higher probability of observing a similar variation level, and random variations causing random differences between various devices within a die.

To measure the impact of process variations on the delay and leakage of our cache model, we considered 5 different variation parameters. These are metal thickness (T), inter-layer dielectric thickness (ILD or H), line-width (W) on interconnects, gate length (L_{gate}), and threshold voltage (V_t) for the MOS devices. We picked separate values for T, H, W, L_{gate}, and V_t for the decoder, pre-charge circuits, memory cell arrays, sense amplifiers, and output drivers of each cache, using the variation limits given by Nassif [15]. Similarly, various parameter values are selected for the remaining components. The mean and 3σ values for each source of variation are listed in Table 1.

Table 1. Nominal and 3σ variation values for each source of process variations modeled

	Gate Length (L_{gate})	Threshold Voltage (V_t)	Metal Width (W)	Metal Thickness (T)	ILD Thickness (H)
Nominal Value	45 nm	220 mV	0.25µm	0.55µm	0.15µm
3σ - Variation [%]	±10	±18	±33	±33	±35

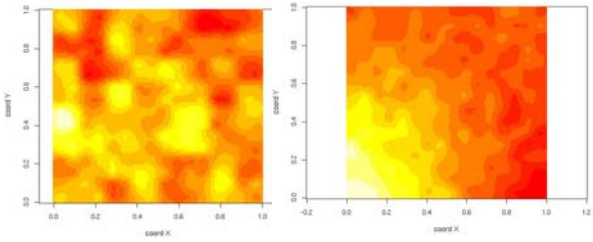

Figure 3. Maps showing the variation of threshold voltage for different range parameters: φ = 0.3 (left) and φ = 0.5 (right).

We model both systematic and random process variations for our processor model. To take into account the spatial correlation we use a range factor (φ) in the two dimensional layout of the chip. Thus, each process parameter can be expressed as a function of its mean (µ), variation (σ), and the range (φ) values. If two points x_i and y_i in a 2D plane are separated by a distance d_i, then the spatial correlation factor between them can be described as an inverse linear function involving φ and d_i. Note that there is no correlation between two spatial points, which are φ units or more apart. In addition to the spatially-correlated variations, we also model random variations in the process parameters. To model them, we chose process parameters randomly from a uniform distribution. Spatially correlated process variations are found to be the dominating factor [10]. In addition, the contribution of the random variations to the overall variations changes according to the parameter [10]. Therefore, we have set lower levels of random variations compared to systematic variations and generally the amount of random variations do not exceed 30% of the total variation. With this background, we have generated a spatial map of various parameter values using the R statistical tool [1]. Figure 3 shows the different threshold voltage maps for φ set to 0.3 and 0.5. We must note that φ value has a considerable impact on the randomness of the parameter values. A higher φ means that the values are highly correlated, whereas a low φ value results in a highly random parameter value distribution. To extract the parameter values corresponding to the different functional units, we use the floorplan of Alpha EV7 processor. In other words, the process variation values for the chip were generated first, followed by the extraction of the values that correspond to the particular positions of the studied components from this modeled chip.

3. Modeling Speed-Binning

In order to effectively estimate the binning distribution and demonstrate the effect the process variations on it, we chose a set of 1100 chips for our analysis. Using the process parameters described in Section 2, their delay and leakage current values are obtained from SPICE simulations for the cases when φ=0.3 and φ=0.5, which in turn are used to determine the binning and yield loss. The cut-off for delay has been set to be the sum of the mean and standard deviation of the delay of the simulated chips (µ + σ), whereas the leakage cut-off has been set to be three times the mean leakage value. These limits are based on previous studies [20].

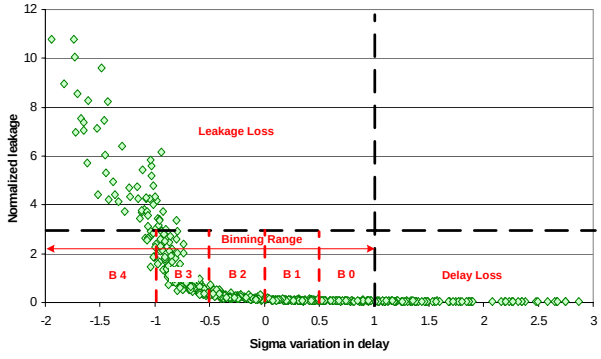

Figure 4. Normalized leakage and delay distribution scatter plot for simulated chips showing the binning for 5-bin strategy. B0 through B4 represent the bin numbers from lowest to highest frequency

Most processor families are available in discrete frequency intervals. For example, the frequency for the Intel Pentium 4 processor family starts with 3.0 GHz and reaches 3.8 GHz with equal intervals of 0.2 GHz [11]. Similarly, other commercial processors by AMD, Intel, and Texas Instruments (TI) are marketed with 5 or 6 different frequency ratings. Our binning methodology also assumes equal binning intervals. This interval is chosen depending on the number of bins to be generated. Regardless of the number of bins, any chip that has a delay greater than the '$\mu + \sigma$' limit is referred to as a delay loss. Chips that satisfy this criterion are used for binning into discrete bins starting from the slowest to the fastest bin. Within each bin, the chips that are lost due to excessive leakage (exceeding the limit of 3x mean leakage) are referred to as the leakage loss. Figure 4 shows the distribution of the normalized leakage power consumption versus the distribution of processor latencies for the base case (i.e., without any architectural optimizations) for the 1100 simulated chips for φ value of 0.5. It also shows the binning for a strategy that generates 5 distinct bins. In this case, the chips that lie within '$\mu + \sigma$' and '$\mu + 0.5\sigma$' delay values are put into Bin0 (denoted by B0 in Figure 4). These correspond to the slowest chips. Similarly, chips with latencies within '$\mu + 0.5\sigma$' and 'μ' are assigned to Bin1. The intervals for the remaining bins are set following the same '0.5σ' interval. Note that the highest bin consists of the chips with delay values less than '$\mu - \sigma$'. Using a similar methodology, we model a strategy that generates 6 bins. In this case, we reduce the binning interval to '0.4σ'. Hence, Bin0 consists of chips that fall between '$\mu + \sigma$' and '$\mu + 0.6\sigma$', Bin1 consists of chips that fall between '$\mu + 0.6\sigma$' and '$\mu + 0.2\sigma$', and likewise.

4. PROPOSED CACHE ARCHITECTURES

In the first part of this section, we first describe cache resizing schemes One-Way Sizing (OWS) and Multi-Way Sizing (MWS). In Section 4.2, we describe a novel cache architecture called Substitute Cache (SC), which masks the effects of process variations by including extra storage in the cache. These schemes aim at increasing the number of chips in the higher frequency bins and hence improve the average performance of the manufactured chips.

4.1 Cache Resizing Schemes

The main idea in these schemes is to analyze the design of the cache, determine the word lines that can cause a delay violation and then modify the cache architecture such that these word lines may be disabled. In the core of these ideas lies one common characteristic: if a path is found to be a critical path in a cache architecture, it will be the critical path in a large number of chips. In general, when process variations are considered, it is hard to determine a single path that is the critical path. Therefore, each path is associated with a probability of being a critical path. If this probability is *X*%, the corresponding path is expected to be the critical path in *X*% of the manufactured chips. In our cache model, we have observed that these probabilities can be very high. Particularly, our analysis of the cache architecture and the process variation simulations (see Section 2) reveal that one particular word line is the critical path in 67.3% of the 1100 caches we have studied. The reasons for this phenomenon are two-fold. First, cache architectures are regular; most paths exhibit the same characteristics. Second, because of spatial correlation in process variations, all the word lines are affected similarly. The consequence of this phenomenon is crucial: if we select a word line to be the critical path during the design process, it will be the critical path in many chips and hence disabling it may reduce the overall cache delay.

4.1.1 One-Way Sizing (OWS)

As the name suggests, One-Way Sizing (OWS) refers to the cache resizing scheme when resizing is restricted to a single cache way. The main idea in OWS is to disable word lines that are likely to generate cache delay violations or cause the chip to be placed in the lower bins. Take for example the 4-way set associative cache described in Section 2.1. If due to the effects of process variations the incurred extra delay makes the cache very slow, then turning off the delay-intensive line will be helpful in decreasing the cache latency. As a result, the chip can be placed in a higher-frequency bin. Our OWS scheme is based on this concept. Particularly, we first analyze the cache architecture and determine the critical paths. Each critical or near-critical path corresponds to a word line. Then, we select ***n*** such paths and change their word line select bit logic to allow the designer to disable them (i.e., turn them off). The number of cache rows or word lines to be disabled depends on the cost and overhead the designer is willing to allow. For example, OWS-4 refers to disabling the 4 most critical word lines of the cache. Note that, to simplify the process of disabling, we do not allow each line to be turned on/off individually. On the contrary, all the selected lines are enabled/disabled together. To clarify the process, consider the process of developing the OWS-8 scheme. For OWS-8, we first analyze the delay of all word lines in a cache way. In our cache architecture, there are 128 such lines; hence, we order them according to their expected latency. Then, we choose the topmost 8 and change their word line select logic. This can be performed by adding an additional input to the AND gates that activate the local word line select signals. This additional input is used for the enable signal. The enable signals for all the 8 word lines are connected to the same "resize enable" signal. After the manufacturing, using this enable signal, the designer can choose to disable all the selected 8-rows. If one of these word lines is the critical path, the total delay of the cache will be reduced. As a result, the chip may be placed in a higher bin.

Note that, in OWS, each cache way has a separate "resize enable" signal. As a result, the speed-binning process after the manufacturing needs to be changed to test the overall delay while each of these signals is asserted. Although it is possible to control each enable signal (and hence the cache way) individually, the number of possible

combinations can be large. In addition, if several word lines corresponding to the same index are disabled, the associativity for those indexes may decrease, potentially resulting in a large number of cache misses. Therefore, we allow at most one set of disabled words lines. In other words, only selected word lines from one cache way can be disabled at a given time. To implement this, each "resize enable" signal will be asserted sequentially during the testing stage, and one signal will be allowed to remain high if this changes the outcome of the speed-binning.

4.1.2 Multi-Way Sizing (MWS)

OWS aims to locate the likely critical paths in a cache and embed enable/disable signals for them, so that these word lines can be disabled. However, if a word line is the critical path in one of the cache ways, it is very likely to be the critical path in the remaining ways. As a result, although OWS can disable one of these paths, the remaining ones will still be enabled and cause a long cache access delay. Another drawback of the OWS scheme is the increased complexity due to the "resize enable" signals in each way. The Multi-Way Sizing (MWS) technique aims at attacking these limitations. Particularly, MWS disables all the chosen critical word lines from all the cache ways instead of disabling the word lines in a single cache way as done in OWS. To explain the idea, consider that word line N is determined to be the most likely critical path. Then, similar to the OWS scheme, MWS will change the AND gates on word line N to allow it to be disabled. However, unlike OWS, MWS will allow the designer to disable all word line N's from all the cache ways simultaneously. If the word line N is the critical path in all the cache ways, this will eliminate the longest path in each way and cause a significant reduction in the cache delay. Because of the spatial correlation of process variations, the probability that the same index remains the critical path in different cache ways is high; in these cases MWS improves upon OWS.

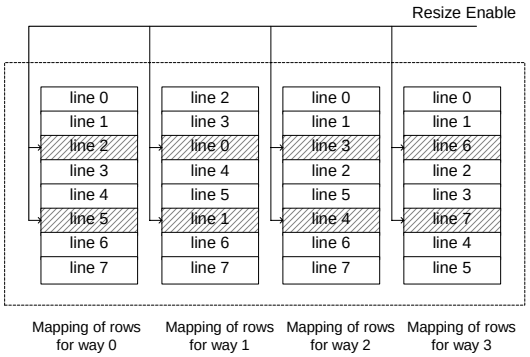

Figure 5. The mapping of indexes to word lines for MWS. Straddled blocks show the lines disabled by the "resize enable" signal.

A second advantage of the MWS scheme is the reduction in the number of enable/disable signals. Since the decision of enabling/disabling is done for the whole cache, the cache will implement a single "resize enable" signal as opposed to one for each way in the case of OWS. This will reduce the complexity of the control circuitry.

For MWS, similar to OWS, the designer has to select the number of rows that will implement the enable/disable signals. If 4 word lines from each cache way are connected to the "resize enable" signal, the scheme is called MWS-4. Note that, this corresponds to disabling 16 word lines simultaneously for a 4-way cache.

A problem with the MWS technique is that when a cache line is disabled across all the ways, that index loses its address space. For the above-presented example, if we decide to disable all the word lines N, then any addresses with the corresponding index will miss in the cache. To tackle this issue, the orientations of the decoder lines are changed in such a manner that no identical indexes are disabled in two different ways. To be precise, we modify the mapping of indexes to word lines in each cache way such that each index can be disabled at most once. Figure 5 presents the change of the mapping for a 4-way, 32-entry cache (8-entries for each way). The initial word lines to be disabled are found using the delay analysis. In our example, these are lines 2 and 5. Then, for the remaining cache ways, the rows to be disabled are found by considering the lowest row number that has not yet been placed into a disabled line. In our example, these are lines 0 and 1 for cache way 1, lines 3 and 4 for cache way 2, and lines 6 and 7 for cache way 3. The remaining rows are mapped to remaining index numbers in order. As a result of this reordering, when the cache is resized, the associativity for each index reduces by at most one. Particularly, for our example architecture, each index has exactly 3 enabled rows, hence, the cache miss rate will be identical to that of a 3-way associative cache with 24 total entries.

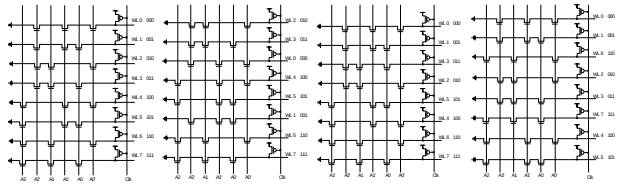

Figure 6. Post-decoder implementation for changing the index to word-line mapping.

This remapping of the indexes to word lines can be implemented by changing the post-decoders that are implemented in high-performance caches. Particularly, the decoders in modern caches work in two stages: a pre-decode and a post-decode stage. The pre-decode stage generates a number of signals and broadcasts them to each word line, where the post-decoders are waiting for certain combinations. The new mapping of the indexes to word lines can be implemented by simply changing these combinations. The post-decoder implementations for the cache architecture shown in Figure 5 are depicted in Figure 6. The signals A0, A0', A1, A1', A2, and A2' are produced by the pre-decoder. The select logic (i.e., transistors on the word line select logic) for each word line corresponds to the post-decoder stage. As shown in the figure, by simply

reordering the locations of these transistors, we achieve the desired reordering. Note that this change does not incur any penalty on the delay of the cache.

4.1.3 Complexity of Resizing

Both our MWS and OWS schemes have design overheads. Note that we implement the enable signals on the critical paths of the cache; hence any change, due to our schemes, increases the cache delay. The particular modification we make to the cache is to change the 2-input AND gate that enables the word line select signal to a 3-input AND gate. We found that the delay overhead for this extra circuitry is 0.75% on average. This increase in delay has an impact on the binning of the chips. However, we must note this overhead is not applicable to MWS when these lines are disabled. To be precise, when the selected word lines are disabled, they will never be used throughout the lifetime of the chip. Therefore, the delays of these lines are not considered during the critical path analysis, hence the overall delay is not affected for MWS. For OWS, on the other hand, this increase in delay may have an impact on the cache delay. Since OWS disables word lines in only one of the cache ways, the delay of the word lines in the remaining cache ways may increase, which in turn will increase the critical path delay.

4.1.4 Effects of MWS and OWS on Cycles-per-Instruction (CPI)

Since we are performing cache resizing, our schemes may increase cache miss rates, which will result in performance degradation. In this section, we analyze how our schemes change the cycles-per-instruction (CPI) for the SPEC2000 applications. SimpleScalar 3.0 [23] simulator is used to measure the effects of our proposed cache resizing techniques. The necessary modifications have been implemented on the base simulator to model selective cache replay, the buses between caches, and port contention on caches. Changes were also made to SimpleScalar to implement the cache resizing schemes, which disable certain indexes from corresponding cache ways. The base processor is a 4-way processor with an issue queue of 128 entries and a ROB of 256 entries. The simulated processor has disjoint level 1 data and instruction caches: level 1 data cache is a 32 KB, 4-way set associative cache with block size of 64-bytes and latency of 4 cycles, and the level 1 instruction cache is a 32 KB 4-way set associative cache with block size of 32-bytes and latency of 2 cycles. The level 2 cache is a unified 1024 KB, 8-way set associative are cache with 128 byte block size and 20 cycle latency. The memory access delay is set to 350 cycles. We have performed our simulations using 11 floating point and 12 integer benchmarks from the SPEC2000 benchmarking suite [26].

Figure 7 and Figure 8 present the increase in CPI for the MWS and OWS schemes, respectively. The average increase in the CPI is 0.08% for MWS-8 and 0.02% for OWS-8 schemes. Among the studied applications, only two exhibit an increase in CPI exceeding 0.3%: gzip and apsi. For these applications, the increases in CPI for the MWS-8 scheme are 0.55% and 0.32%, respectively.

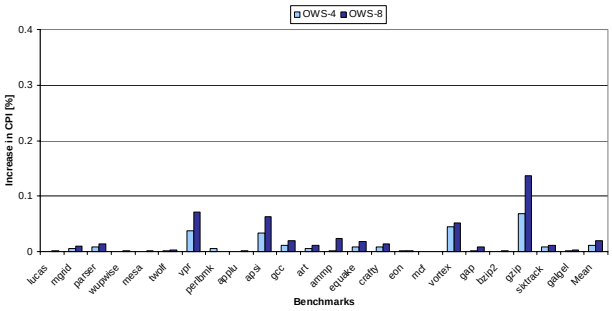

Figure 7. Performance results for OWS schemes for the SPEC2000 applications

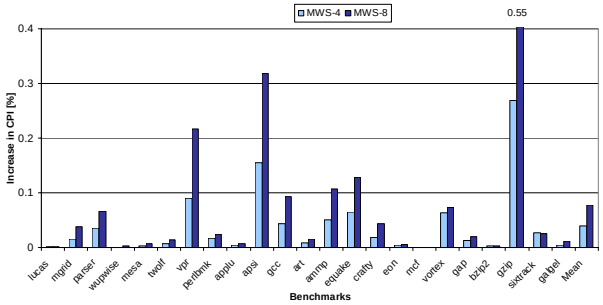

Figure 8. Performance results of MWS schemes for the SPEC2000 applications.

Note that the schemes disable sporadic indexes, and hence different indexes have varying associativities, creating heterogeneous cache architecture. Therefore, the increase in the CPI for these two applications is directly caused by their usage of the disabled indexes.

4.2 Substitute Cache

The main downside of the schemes discussed in previous section is the performance impact of resizing the cache in terms of increased CPI levels. Our third scheme named Substitute Cache (SC) attacks this problem. The idea in the SC is to augment each cache way with extra storage that will be used if certain locations in the main cache exhibit long latencies. In such cases, the data will be read from the substitute cache, and chips from the lower frequency bins can now be placed in higher frequency bins, because the high latency lines are not used. Moreover, some of the chips, which could have failed due to high access latencies, will be added to the overall yield.

The anatomy of the proposed cache architecture is shown in Figure 9. For the sake of clarity we detail the use of SC on a single cache way; however, each cache way has a similar SC associated with it. SC is similar to a fully-associative cache structure. In our study, its size is either 4 or 8 entries. As opposed to the L1 cache, SC has smaller line sizes. Particularly, it consists of only 64-bit entries, because it stores words of the main data array. Instead of storing the whole cache line, only the critical word in the

line is stored in the SC, because our study reveals that the words with maximum access latency are always the ones that are furthest from the decoder. As a result, by just storing these words, we obtain the same improvement in cache frequency while keeping the SC size small. However, if necessary, words in other locations can also be placed into the SC. An SC is divided into 2 components: an index table and a data array. Note that the SC uses the column multiplexers and output drivers of the main array. Whenever a cache word is placed in the data array of the SC, index bits of its address, which is equal to the sum of the row and column addresses (10 bits in our architecture) are placed in the index table of the SC. For example, if we decide to place the word with index value 0x044 to the SC, we will have an entry in the index table with value 0x044. Note that this word would have resided in the row with index 0x8 in the main array with the column address being equal to 0x4. In case of a data access, the index table is checked with the index bits of the address. A match implies that the data will be read from the SC instead of the main array. Specifically, if the index of the address is found in the SC index table, the contents of the corresponding data array row are forwarded to the column multiplexers of the main array. The additional control logic shown in Figure 9 will then set the column multiplexers correctly. If the index of the address does not match any index table entries, the main array will be accessed. Even if there is a match in the index table, the access can still miss in the cache if the corresponding tag does not match. However, the tag structure is not affected by the addition of SC. If there is a miss due to tag mismatch, we will still output the data, which will be ignored because the tag will indicate the miss. Overall, the tag match/mismatch is independent of the SC design. We only care whether the corresponding parts of the address match with the values stored in the index table and decide whether to supply the data from the main array or the SC.

Now let us consider a typical read operation in the main array. The row address part of the index field selects the appropriate row in the data array through the row decoder. The appropriate word is then chosen by the column multiplexers with the help of the column address bits of the index. One of the key observations is the difference between the time taken by each of these steps. Particularly, the inputs to the column multiplexers are available at the same time the decoder is accessed. However, the signals provided to the decoders will traverse through the decoder logic, the word lines, the memory cell, the bit lines, and the sense amplifiers before it will reach the column multiplexers. We utilize this imbalance to operate our SC structure. As soon as the address is available, we start accessing the SC index table. If they record a hit, we change the input to the column multiplexers to 0. In other words, we forward the output of the SC as the output of the cache. If, on the other hand, there is no match in the index table, we will set the column multiplexer to the original position indicated by the column address. If the time to check the index table in the SC is less than the delay of the data array (the sum of the delays of the decoder, word line, memory cell, bit line, and sense amplifier), then, this operation does not cause any delay overhead on the cache, because while the data array is accessed, we would have already determined the hit/miss in the SC index table. Using CACTI 3.2 [22] we found the total access latency for a 8-entry SC to be 0.28 nanoseconds; whereas the latency for the main array (one set of the 32KB 4-way set associative cache) is 0.40 nanoseconds. Therefore, the SC access can be completely overlapped with the main array access and will not cause any increase in the cache access latency. The only change in the latency of the main array is due to the changes in the column multiplexers. Because of the data forwarding from the SC, the column multiplexers (straddled in Figure 9) have an additional input coming from the SC data array. The analysis with our SPICE model reveals that this overhead is 0.34% of the overall cache access latency. We include this overhead during our binning analysis in Section 5.1. Note that ***there is no performance loss in terms of CPI for the SC scheme,*** as the effective cache size remains unchanged.

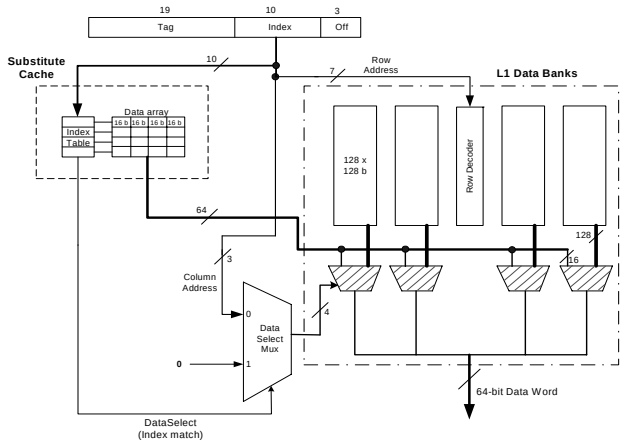

Figure 9. One cache way of a 32KB 4-way set associative L1 cache augmented with Substitute Cache. Column muxes are shaded as they select data from 9 inputs as opposed to 8 inputs.

Similar to a read operation, a write access (either a store operation or write operation during the replacement of a cache line) selects the appropriate index using the row and column addresses and updates the selected word in the cache way selected by the Way-Select Logic. In case of the SC, the index table is checked for the index of the data word to be written. If there is a match, the new data word is loaded in the data array of the SC.

One of the key components during the operation of SC is the index table. After the chip is manufactured, a Built-In-Self-Test (BIST) is performed where ***n*** most critical cache indices are chosen and placed in the SC index table. Note that, these values are extracted only once during the lifetime of the chip and never changed. Therefore, they can be extracted by the BIST and become part of the booting

process, where they are read from a permanent location and placed into the index table every time the processor boots. We must mention that we neglect any impact of process variations on the SC since it is much smaller than the L1 cache and its latency is significantly lower, hence it is unlikely to become the critical component. Finally, it should be noted that the size of the SC dictates the area and power overhead of this approach. With the help of SPICE and CACTI, we found the total power overhead to be 6.0% and 6.5% of the main array for a 4- and 8-entry SC, respectively. The area of the cache increases by 3.7% and 4.1% for the 4- and 8-entry SC, respectively.

Overall SC scheme chooses the lines *dynamically* so it is not based on any variation model, as compared to cache resizing schemes rely strongly on the systematic variations.

5. EXPERIMENTAL RESULTS

In this section, we describe the analysis of our proposed schemes. Section 5.1 describes how our schemes change the outcome of the speed-binning, whereas Section 5.2 illustrates the gain in batch-performance.

5.1 Binning Results

This section presents the binning results based on the binning methodology described in Section 3. Since our binning schemes are divided into two categories, namely 5-bin and 6-bin strategies, we describe them separately. For both 5-bin and 6-bin strategies, the proposed MWS, OWS, and SC schemes are applied and the resulting changes in the number of chips in each bin are found. To find how the chips are placed into different bins, we first analyze our architecture with the base cache and find the mean and standard deviation of the 1100 cache delays. Then, based on these values, the boundaries for each bin are set. We then apply the MWS, OWS, and SC to find the new delays for each chip and find the corresponding bin distribution.

Figure 10 and Figure 11 show the binning results for 5-bin strategy for MWS and OWS, respectively. The results for the 6-bin strategy were similar and hence are not presented in detail; in the next section we present the overall impact of the schemes for the 6-bin strategy. To understand the figures, consider the leftmost bar for each bin. This bar corresponds to the number of chips in that bin for the base cache architecture. The bars next to it (i.e., the ones in the middle) represent the number of chips in that bin when MWS-4 or OWS-4 schemes are applied. The right bars represent the number of chips in the corresponding bin for the MWS-8 or OWS-8 schemes. In general, we see that our schemes can successfully increase the number of chips in the higher bins. For example, in the 5-bin strategy, the number of chips in the highest bin (Bin_4) is increased by 8.2% using MWS-8. Figure 12 depicts the binning results for the SC scheme. In case of the SC, the chip yield is catapulted to a larger extent (14.4%). Like MWS and OWS, it also shows a sharp increase in the chip of the last bin for the 5-bin strategy.

It is misleading to draw any conclusion about high-frequency chip yield by simply considering the chips in the highest bin. The gain in the highest bins for all the 3 schemes are accompanied by a reduction in the number of chips in the lower bins. However, we must note that the total yield is increased using these schemes. Specifically, the total yield increases by 4.5%, 3.5% and 9.7% using the MWS-8, OWS-8, and SC-8 schemes, respectively (for φ=0.5). Although there are no additional chips lost due to leakage for the resizing schemes, the SC is associated with a power overhead. The SC-4 and SC-8 schemes cause an additional 9.1% and 11.7% loss of chips, respectively. In spite of that, the total yield increases for SC, because it converts a high number of delay loss chips into yield. Even though the total number of chips increases, the schemes tend to move a larger number of chips towards the higher bins. As a result, the chip counts in the lower bins tend to decrease.

One of the reasons for the significant change of yield gain from MWS-4 to MWS-8, OWS-4 to OWS-8, and SC-4 and SC-8 is the fixed cost of implementing the schemes. As described in Section 4.1, implementing the resizing scheme incurs a circuit delay of 0.75% over the base cache architecture. When the "resize enable" signal is off, this overhead in delay is added to the critical path, whereas when it is on it does not affect the critical delay in the MWS scheme. Therefore, MWS has a more profound impact on the speed-binning outcome. In case of OWS, the overhead may cause other cache ways to become the critical path, limiting its overall impact. For SC, this overhead is even lower and hence it achieves better binning results than MWS.

5.2 Impact on Batch-Performance

To summarize the effect of the new binning distribution and to judge its merits, we define a new metric called ***batch performance (BP)***. Batch performance corresponds to the total performance of the chips obtained from a batch of microprocessors. If there are k different frequency bins having frequency ratings $f_1, f_2,\ldots, f_k$ with each of them having yields $n_1, n_2,\ldots,n_k$; the total batch-performance is given by:

$$BP = \sum_k (f_k \times n_k) \quad (1)$$

This BP formula can be is extended in two ways. First, if an architectural scheme has an impact on the CPI, the change can be captured by incorporating it into the equation. Specifically, if a scheme achieves an IPC of $i_1, i_2,\ldots, i_k$ for each bin, the new batch performance will be calculated by:

$$BP = \sum_k (f_k \times n_k \times i_k) \quad (2)$$

Finally, to find the average performance, this sum is divided to the number of manufactured chips. We have calculated the average BP for the base cache architecture and our proposed schemes based on Equation 2. Table 2 presents the change in BP with our architectural schemes.

As mentioned in Section 4.1, the MWS and OWS schemes introduce some performance overhead in terms of increase in CPI. This means that for these schemes, the effective IPC goes down and thus changes the batch

performance. For example, increase in CPI by 0.08% for MWS-8 changes the effective IPC to 0.999 for the same (assuming base IPC to be 1). Thus the BP for MWS are calculated using the changed IPC. The SC scheme, however, has no effect on the IPC and thus the enhancement in binning distribution is directly converted into batch performance improvement.

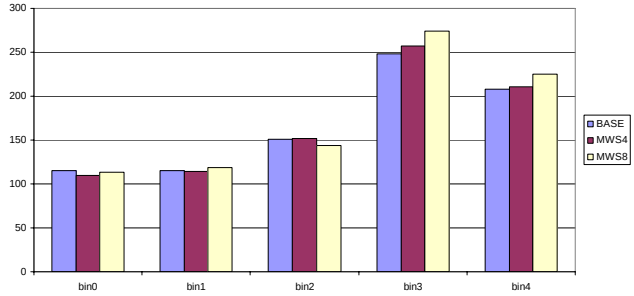

Figure 10. Binning with 5-bin strategy for MWS.

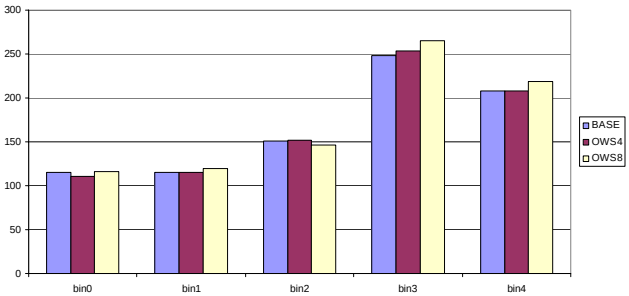

Figure 11. Binning with 5-bin strategy for OWS.

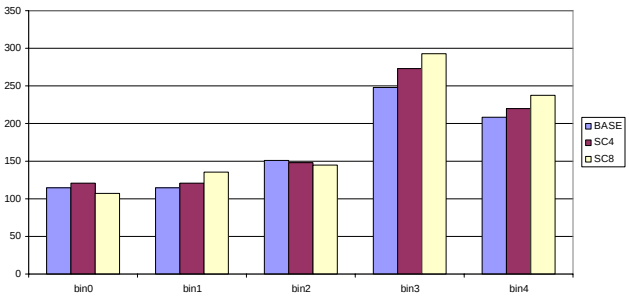

Figure 12. Binning with 5-bin strategy for SC

Table 2. Increase in batch performance for various cache-architectures

Range (φ)	Binning Strategy	Increase in Batch Performance with respect to the base architecture [%]					
		MWS		OWS		SC	
		4	8	4	8	4	8
0.5	5-bin	1.35	5.71	0.59	4.09	5.88	11.50
	6-bin	1.30	5.83	0.71	4.21	5.58	11.19
0.3	5-bin	1.95	5.26	1.35	4.08	6.18	10.51
	6-bin	2.10	5.60	0.98	4.32	6.10	11.59

An important point to note here is the relative batch performance improvement of OWS-8 with respect to MWS-4. Although the latter shuts off 16 lines compared to 8 lines in OWS-8, MWS-4 has a lower BP improvement than OWS-8. The main reason behind this is the high spatial correlation within a particular cache way, which restricts the criticality of a cache to a single way. In other words, we observe that different cache paths are affected in a similar manner under process variations. As a result of the correlation of process variations, these changes result in one cache way containing several critical or near-critical paths. Therefore, OWS becomes a more effective scheme than MWS, as shutting off lines within a way is preferable to spanning them across multiple ways. Overall, SC-8 scheme performs the best, improving the batch performance by 11.5% and 11.2% for the 5-bin and 6-bin strategies under φ=0.5, respectively. For φ=0.3, the improvement for the same binning strategies are 10.5% and 11.6%, respectively.

We must note that from a manufacturer's perspective, there is a strong motivation to increase the batch performance. Assuming a simplistic case, a higher BP can translate into higher revenue: the high-frequency chips are sold for a higher price, hence an increase in BP will result in revenue increase. Overall, since the manufactured chips will have varying performance levels, it is logical to consider the overall performance of the chips rather than the possible enhancements to a single processor.

6. RELATED WORK

Mitigating the effects of process variations has long been the objective of circuit designers. Previous works show that several circuit-level techniques have been adopted to counter the negative effects of process variations [4, 6, 9, 18, 21, 24]. The inter-die and intra-die process variations and their effects on circuit leakage is studied in detail by Rao et al. [19]. In another work, Rao et al. [20] analyze the impact of process variations on circuit leakage and proposed methods to reduce them. Most of these techniques focus on analyzing the design statistically or by using static timing analysis, and then modifying the parts of the circuits that are most susceptible to variations. Many gate-sizing strategies have been used on the critical or near critical regions of the circuit in order to reduce the effective latency [7, 28].

Performance binning has also been adopted as means of increasing yield [4, 8, 21]. Datta et al. [8] propose a novel approach of changing the effective speed-binning by gate sizing, and thus increasing the profit. Unlike our schemes, most of their analyses are based on statistical estimations of yield, and the optimizations are for high-level synthesis. Kim et al. [12] have studied the effects of cache size on leakage and analyzed the trade off on access time when multiple threshold voltages are used for level 1 and level 2 caches.

In the architecture domain, system-level techniques are studied to prevent yield loss under process variations [17]. Sohi's work show that cache redundancy can be used to

prevent yield loss [25]. At a high level, SC resembles their cache duplication scheme. However, there are several differences in our implementation. Most importantly, rather than implementing a separate structure, we extend the main cache and hence can utilize many structures of it. Ozdemir et al. [17] present yield-aware microarchitectural schemes specially in cache domain, that improved the overall yield to as much as 97%. Liang et al. [13] target at mitigating the effects of process variation by introducing variable latency regular structures like register files. Besides, Agarwal et al. [2] propose a scheme that prevents yield loss due to failures in the SRAM cells of the cache. Their approach is mostly based on Built-In Self-Test (BIST) circuitry and the cache optimizations are concentrated towards yield maximization. In comparison to the abovementioned works, our efforts have been directed towards effective binning and batch performance optimization for set associative caches. In addition, all the previous techniques listed above have performance implications, i.e., different chips from the same family may exhibit varying performance levels. However, our SC scheme provides constant IPC for all the chips in a family.

7. CONCLUSIONS

Efficient binning under process variations is becoming a significant challenge for chip manufacturers. A considerable amount of effort is being made to save chips from excessive delay and market them properly after speed-binning. In this paper, we studied two cache architectures, which are aimed at maximizing the batch-performance of a particular line of chips manufactured with the same process technology. Our first scheme, One-Way Sizing (OWS), tries to resize a single way of a set associative cache for reducing cache latencies and hence improving the frequency. The second approach called Multi-Way Sizing (MWS) extends this concept to multiple cache ways. The extra circuitry needed for these schemes is very small and the newly resized cache causes minimal reduction in the instruction-per-cycle (IPC) rates: 0.02% and 0.08% on average for the most aggressive OWS and MWS, respectively. As an alternative to these resizing schemes, we propose a novel technique called Substitute Cache (SC), which has no performance overhead and works by storing critical words of the data array in a separate structure. Overall, the most aggressive OWS, MWS and SC schemes increase the average batch performance by 4.2%, 5.8% and 11.6%, respectively.

REFERENCES

[1] "The R Project for Statistical Computing", Available at http://www.r-project.org/

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

[3] B. S. Amrutur and M. A. Horowitz, "Speed and Power Scaling of SRAM's," *IEEE Trans. on Solid-State Circuits*, vol. 35, pp. 175-185, Feb. 2000.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter Variations and Impact on Circuits and Microarchitectures," In Proc. of *Proc. of the Design Automation Conference*, 2003.

[5] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," In Proc. of *Custom Integrated Circuits Conference*, 2000.

[6] S. H. Choi, B. C. Paul, and K. Roy, "Novel Sizing Algorithm for Yield Improvement Under Process Variation in Nanometer Technology," in *Proc. of the Design Automation Conference*. San Diego, CA, 2004, pp. 454-459.

[7] O. Coudert, "Gate Sizing: A General Purpose Optimization Approach," In Proc. of *European Design and Test Conference*, 1996.

[8] A. Datta, S. Bhunia, J. H. Choi, S. Mukhopadhyay, and K. Roy, "Speed Binning Aware Design Methodology to Improve Profit Under Parameter Variations," In Proc. of *Proc. of the Conf. on Asia South Pacific Design Automation*, 2006.

[9] A. Datta, S. Bhunia, S. Mukhopadhyay, and K. Roy, "Delay Modeling and Statistical Design of Pipelined Circuit Under Process Variation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2006.

[10] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos, "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization," In Proc. of *Proc. of the Intl. Symposium on Quality of Electronic Design*, 2005.

[11] Intel, "Intel Processor Pricing", 2006, Available at http://www.intel.com/intel/finance/pricelist/processor_price_list.pdf?iid=InvRel+pricelist_pdf

[12] N. S. Kim, D. Blaauw, and T. Mudge, "Leakage Power Optimization Techniques for Ultra Deep Sub-Micron Multi-Level Caches," In Proc. of *International Conference on Computer Aided Design*, 2003.

[13] X. Liang and D. Brooks, "Mitigating the Impact of Process Variations on CPU Register File and Execution Units," In Proc. of *International Symposium on Microarchitecture*, 2006.

[14] M. Miller, "Manufacturing-aware Design Helps Boost IC Yield", Sep. 2004, Available at http://www.eetimes.com/news/design/features/showArticle.jhtml;?articleID=47102054

[15] S. R. Nassif, "Modeling and Analysis of Manufacturing Variations," In Proc. of *IEEE Conference on Custom Integrated Circuits*, May 2001.

[16] S. Natarajan, M. A. Breuer, and S. K. Gupta, "Process Variations and their Impact on Circuit Operation," In Proc. of *International Symposium on Defect and Fault Tolerance in VLSI Systems*, 1999.

[17] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, "Yield-Aware Cache Architectures," In Proc. of *International Symposium on Microarchitecture*, 2006.

[18] S. Raj, S. B. K. Vrudhula, and J. Wang, "A Methodology to Improve Timing Yield in the Presence of Process Variations," In Proc. of *Proc. of the Conf. on Design Automation*, 2004.

[19] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process Variation," In Proc. of *ISLPED '03*, 2003.

[20] R. R. Rao, D. Blaauw, D. Sylvester, and A. Devgan, "Modeling and Analysis of Parametric Yield under Power

and Performance Constraints," *IEEE Des. Test*, vol. 22, pp. 376-385, 2005.

[21] A. Raychowdhury, S. Ghosh, S. Bhunia, D. Ghosh, and K. Roy, "A Novel On-chip Delay Measurement Hardware for Efficient Speed Binning," In Proc. of *Intl. Online Testing Symposium*, Jul. 2005.

[22] P. Shivakumar and Norman Jouppi, "CACTI 3.0: An Integrated Cache Timing, Power, and Area Model," WRL Research Report 2002.

[23] SimpleScalarLLC, "The SimpleScalar Tool Set," 2001.

[24] D. Sinha, N. V. Shenoy, and H. Zhou, "Statistical Gate Sizing for Timing Yield Optimization," In Proc. of *Proc. Intl. Conf. on Computer-Aided Design*, 2005.

[25] G. S. Sohi, "Cache Memory Organization to Enhance the Yield of High Performance VLSI Processors," *IEEE Trans. Comput.*, vol. 38, pp. 484-492, 1989.

[26] SPEC, "Spec CPU2000: Performance Evaluation in the New Millennium v1.1," Dec. 2000.

[27] Sun, "OpenSPARC T1", Available at http://opensparc-t1.sunsource.net/index.html

[28] L. Wei, K. Roy, and C.-K. Koh, "Power minimization by simultaneous dual-Vth assignmentand gate-sizing," In Proc. of *IEEE Custom Integrated Circuits Conference (CICC)*, 2000.