Yield-Aware Cache Architectures

Serkan Ozdemir Debjit Sinha* Gokhan Memik Jonathan Adams Hai Zhou
EECS Department, Northwestern University
{soz463, debjit, memik, jra760, haizhou}@ece.northwestern.edu

Abstract

One of the major issues faced by the semiconductor industry today is that of reducing chip yields. As the process technologies have scaled to smaller feature sizes, chip yields have dropped to around 50% or less. This figure is expected to decrease even further in future technologies. To attack this growing problem, we develop four yield-aware microarchitecture schemes for data caches. The first one is called Yield-Aware Power-Down (YAPD). YAPD turns off cache ways that cause delay violation and/or have excessive leakage. We also modify this approach to achieve better yields. This new method is called Horizontal YAPD (H-YAPD), which turns off horizontal regions of the cache instead of ways. A third approach targets delay violation in data caches. Particularly, we develop a VAriable-latency Cache Architecture (VACA). VACA allows different load accesses to be completed with varying latencies. This is enabled by augmenting the functional units with special buffers that allow the dependants of a load operation to stall for a cycle if the load operation is delayed. As a result, if some accesses take longer than the predefined number of cycles, the execution can still be performed correctly, albeit with some performance degradation. A fourth scheme we devise is called a Hybrid mechanism, which combines the YAPD and the VACA. As a result of these schemes, chips that may be tossed away due to parametric yield loss can be saved. Experimental results demonstrate that the yield losses can be reduced by 68.1% and 72.4% with YAPD and H-YAPD schemes and by 33.3% and 81.1% with VACA and Hybrid mechanisms, respectively, improving the overall yield to as much as 97.0%.

1. Introduction

Decreasing yields in modern VLSI chip manufacturing is a critical issue faced by the semiconductor industry. In a drive to continue to meet the demands of Moore's Law, process technology has continually transitioned to smaller sizes with current average feature sizes being as small as 65 nanometers. Although this scaling trend facilitates more gates, and therefore more performance and functionality to be packed onto every chip produced, it has made the manufacturability of these chips increasingly difficult [24, 39]. With process technologies scaling from 350 nanometers to 90 nanometers, chip yields have dropped from over 90% to just above 50% [18]. A recent study on 45 nanometer technologies reports yields around 30% [3]. This trend is depicted in Figure 1, which shows the expected yield for different manufacturing technologies and the factors on which the yield loss is attributed to.

Factors limiting chip yields can be grouped into three categories: defect-density related yield loss, lithography based yield loss, and parametric yield loss. Defect-density related problems are caused by actual errors with the silicon, such as when a contaminating particle is introduced during fabrication. These are well-controlled as silicon and clean-room technology becomes more efficient. Lithography based failures occur when there are defects on the masks used to burn the silicon. These are tied to reticle patterns and are controlled as process technologies mature. Parametric yield loss, on the other hand, occurs because the manufactured chip does not meet a design parameter. For example, a microprocessor, which does not meet a frequency constraint or consumes too much power, may be tossed away.

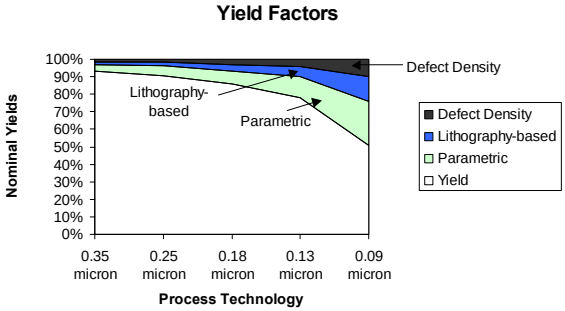

Figure 1. Yield factors for different process technologies [18]

As shown in Figure 1, the impact of all the above factors has worsened with technology scaling. However, parametric losses are the largest inhibitor to chip yields [18] and contribute significantly to overall yield losses starting from the 0.18 micrometer technology generation [1, 10, 16, 24, 25]. For sub-180nm technologies, it becomes harder to control variations in device parameters such as channel length, gate width, oxide thickness, and device threshold voltage. Even in a mature technology like 130nm, these variations are known to cause a 30% variation in maximum allowable frequency of operation, and a fivefold increase in leakage power [10]. For newer technologies, these variations can be even higher: 20X increases in leakage have been

* Debjit Sinha is currently with IBM Microelectronics, USA

reported for 90nm [6]. As a result of this variation in performance and power, an increasingly larger fraction of the manufactured chips do not meet their design constraints and may have to be tossed away, even if they do not contain any functional defects.

Increased power consumption levels is an important factor in reduced yields [26]. At 65nm, sub-threshold leakage power constitutes a higher fraction of total power compared to switching (dynamic) and short circuit power [27]. For each successive technology, this sub-threshold leakage increases fivefold, while gate leakage can increase up to twenty-fold. Only a 10% variation in a transistor's effective channel length causes as much as three-fold difference in sub-threshold leakage current. Gate leakage difference is 15-fold for a similar 10% variation in oxide thickness [32]. Small variations in device threshold voltage result in leakage numbers that differ by a factor of five or ten. This, in turn, increases the total power consumption and causes increased parametric yield loss.

On the other side of the spectrum, delay violations also cause large yield losses. A frequent practice of chip manufacturers is that of frequency- or speed-binning in which they test and qualitatively sort working integrated circuits according to their maximum frequency of operation [10]. The chips that do not meet the very minimum frequency constraints are tossed out during this process. Since circuit delay and leakage current are inversely proportional, this testing also exacerbates parametric yield loss because many dies in the high-speed bin are lost when they exceed acceptable leakage levels, further narrowing the acceptable process window [32].

Yield loss affects both the bottom-line of chip manufacturers as well as the consumers. Every discarded chip increases the cost of those chips that survive the fabrication process. Therefore, it is evident that an effort must be made in future designs to maximize not only performance, but also manufacturability and yields [39]. Our goal in this work is to achieve this by developing microarchitectural techniques to reduce the yield loss. We particularly target the parametric yield loss. Parametric yield loss occurs because a processor does not meet performance and/or power constraints; therefore, it is natural to assume that microarchitectural optimizations can have a significant impact on them. In this work, we develop methods to achieve this and show that parametric yield losses can be significantly reduced.

Rather than trying to apply our ideas to the whole chip, we concentrate on the level 1 data cache. The reasons for focusing on the data cache are three-fold. First, caches consume a relatively large fraction of the processor area and power consumption. Second, level 1 caches have to be built for minimum delay; hence they tend to utilize low threshold voltages. Finally, SRAM structures have high number of independent critical paths and relatively low logic depth in those paths, which makes them the dominant source of unit-to-unit variations [17]. As a result of these three properties, the delay and leakage consumption of level 1 caches change significantly under process variations. Hence, the probability that a chip will not meet the performance/power constraints because of its data cache is high.

To optimize the yield, we aim to reduce losses due to both performance and power constraints. First, we develop the **Yield-Aware Power-Down (YAPD)** technique. Although the notion of YAPD can be applied to different power reduction techniques, in this paper we use a scheme that combines Selective Cache Ways [4] and Gated-Vdd [30] as the example of a yield-unaware power reduction technique. Specifically, we modify this yield-unaware scheme with YAPD to investigate how the yield is affected by our optimization. The main idea in YAPD is to turn off cache ways that cause delay failures. In addition, cache ways can be turned off if they consume excessively large leakage power. We modify this approach to compensate for correlations in process variation parameters. Particularly, we develop the **Horizontal YAPD (H-YAPD)**, which turns off horizontal regions of a cache, instead of a vertical cache way. This optimization reduces the probability of yield loss as we will elaborate further in Section 4.2. By making a small modification in the decoder, we guarantee that the closed regions do not share any common addresses. As a result, for any memory address, the associativity of the cache is identical. Third, we develop the **VAriable-latency Cache Architecture (VACA)**. In a VACA, different cache ways can be accessed with different latencies. The main idea is similar to NUCA [19], however, since we apply the variability to the level 1 data caches, we have to modify the architecture to perform the corresponding instruction scheduling. In addition, we augment the functional units with special buffers that allow an instruction depending on a load operation to stall for a cycle if the load operation is delayed. As a result, if some accesses take longer than the predefined number of cycles, the execution can still be performed correctly, and hence yields are improved over their current levels. Finally, we analyze a **Hybrid** scheme, which combines YAPD (or H-YAPD) and VACA. By combining the advantages of both schemes, the Hybrid scheme achieves the best yield optimizations.

In the next section, we explain how process variations affect yield loss. Section 3 illustrates our cache architecture and our methodology for modeling the process variations on it. Section 4 describes our yield-aware architectures. Section 5 presents the results and Section 6 summarizes related work. We conclude the paper in Section 7 with a summary.

2. Process Variations and Their Effects on Parametric Yield Loss

Process variations can be classified into *inter-die* variations and *intra-die* variations [2, 8]. Inter-die variations denote the variations that occur from one die to the next, from wafer to wafer, and from wafer lot to another. Inter-die variations affect all devices on the same chip similarly, and was considered to be the primary source of process variations in older technology generations [8]. Intra-die variations refer to the variations in device and interconnect

features that are present within a single die (or chip). This implies that two identically designed devices inside a given die may have varying feature dimensions. Intra-die variations are attributed to equipment limitations or statistical effects in the fabrication process, e.g., fluctuations in the doping concentration of devices. With the aggressive scaling of feature sizes in modern technologies, it is natural that this component of process variations is now equally, if not more, significant than the former.

Intra-die variations consist of random and systematic components, thereby producing non-uniform electrical characteristics across a chip [13]. A random variation is defined as the component of a parameter deviation that fluctuates and deviates randomly from device to device, i.e., has zero correlation between devices. A systematic variation is defined as the component of parameter deviation that results from a repeatable and governing principal, where the correlation between devices is empirically determined using a measure of the distance between those devices. This component exhibits strong correlations within a given die. Therefore, ignoring these correlations may cause significant errors in analysis. This *spatial correlation* is locally layout-dependent and circuit-specific, i.e., devices with similar layout patterns and proximity structures tend to have similar characteristics. In addition, the spatial correlation is globally location-dependent, i.e., devices located close to each other are more likely to have the similar characteristics than those placed far away. Designers treat this component of variations as correlated random variables during analysis and optimization. However, some systematic component of intra-die variations, such as Chemical Mechanical Polishing (CMP) and Optical Proximity Correction (OPC) effects, can be directly accounted for during timing or power analysis if parasitic extraction tools can consider their effects.

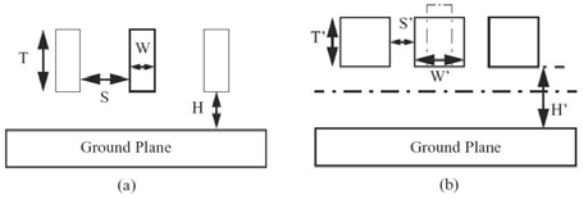

Figure 2. Cross-section of parallel interconnect lines above a ground plane: (a) the ideal case, (b) different types of variations that can exist in the interconnect

Process variations affect both interconnect and devices. Interconnect variations are attributed to those from three components (as shown in Figure 2): metal thickness (T), inter-layer dielectric thickness (ILD or H), and line-width (W or LW) [23]. Additional geometric effects such as sidewall slope or surface and edge roughness may also be of concern, but are not considered in this work. Factors that contribute to interconnect variations include CMP variations from non-uniform metal pattern density, OPC, etching, and electrolytic growth. Note that line-space (S or LS) is not an independent parameter since a variation in line-width automatically causes a change in the line-space. Variation in the interconnect parameters results in a change in its electrical properties, including the resistance (R) and capacitance (C). These electrical parameter variations directly affect signal propagation delays through interconnects, and thereby the performance of a circuit.

Device variations are attributed to variations in gate length (L_{gate}), gate width (W_{gate}), and gate oxide thickness (t_{ox}). Additional sources of variation include those in the drain and source active areas as well as variations in the doping concentration during fabrication. These variations affect the device properties, and thereby, affect circuit performance. The most important sources of device variation are L_{gate}, t_{ox}, and V_t (threshold voltage). Since the ratio of W_{gate}/L_{gate} determines the drain current of a CMOS transistor, if W_{gate} is much larger than L_{gate}, variation in W_{gate} is usually not considered [23].

We next try to qualitatively understand the impact of these variations on latency and power. It is intuitive that the variations in interconnect and device feature sizes contribute to uncertainty in their delays, and therefore, uncertainty in path delays of a circuit. We consider some critical path in a circuit, with the mean of its delay distributions being equal to the required value. The probability that this path satisfies the timing constraint is naturally 0.5 (Parametric timing yield = 50%). If we consider another independent critical path, the probability that the timing constraint of the circuit is met is reduced to $0.5^2 = 0.25$ (Parametric timing yield = 25%). Although critical path delays are correlated in reality, this example gives an intuition of how the variations in device delays contribute to diminishing yields. To study the impact on parametric power yields (that is, the probability of chips that satisfy the power consumption constraints), we separate total power into static and dynamic power. For dynamic power at a specified clock frequency, effective device and interconnect capacitance variations act as the primary source of variability. Next, the sub-threshold leakage is exponentially dependent on the threshold voltage (V_t), which in turn strongly depends on the dopant concentration, and channel length (L_{gate}) [22]. Furthermore, the exponential dependence causes a large spread in the leakage power distribution. For large width transistors, the impact of doping variations on V_t is smaller in comparison to L_{gate}. While low V_t devices are commonly used in circuits to reduce latency, these devices are most vulnerable to high leakage power consumption, leading to large yield loss in high performance bins [6].

Statistical approaches, where the sources of variations are modeled as random variables with known distributions, are considered more suitable for process variation modeling. Analytical approaches to statistical timing analysis have been proposed recently [8, 38], but suffer from inaccuracies due to a large number of assumptions. However, these approaches are efficient and find use in optimization [36]. For accurate analysis, Monte Carlo simulations are widely employed. In this technique, random samples of the random variables are taken in each simulation. The distribution of the final result (could be timing or power) is observed after a large number of Monte Carlo runs have been performed. The advantage is that a realistic distribution is obtained for the

output. The disadvantage is that simulation time can be excessive. In this paper, we also employ a Monte Carlo based simulation framework. Section 5.1 describes how the results of these simulations are used to estimate yield.

3. Cache Architecture and Process Variation Simulations

To be able to estimate the effects of our architectural optimizations on yield, we need to model a cache architecture. To achieve this, we build a HSPICE model for a 16 KB cache that is based on the model by Amrutur and Horowitz [5]. This design is applicable to future generation processors where smaller manufacturing technologies will be used. We use 45 nm PTM technology models [7].

In our model, we implement a 16 KB 4-way set associative cache, where each way is further divided into 4 banks and each memory bank consists of 64x128 bits. To reduce the bitline delays, each bitline is also partitioned into two. Figure 3 shows the details of a single way in our model. 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 driver, parallel wires in decoder, and bit-lines (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. The parasitic values of the interconnect wires are based on the interconnect models from PTM [7]. The gate sizes are then optimized to minimize the cache latency.

Once we have the basic cache model, we modified all blocks to measure the effects of process variations. In our model, we considered variations in metal thickness (T), inter-layer dielectric thickness (ILD or H), line-width (W) on interconnects, and gate length (L_{gate}) and threshold voltage (V_t) for the MOS devices. For a particular cache, we picked different random values for T, H, W, L_{gate} and V_t for the decoder, pre-charge circuits, memory cell arrays, sense amplifiers and output drivers using the variation limits given by Nassif [29]. The mean and 3σ values for each source of variation are listed in Table 1.

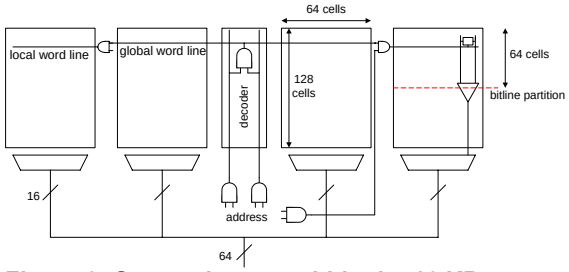

Figure 3. One cache way within the 16 KB, 4 way cache model

To understand the impact of process variations on the cache, we have generated process variation parameters for all the segments of the cache. However, as we have described in Section 2, there is a correlation between entities in a die. Therefore, we use a correlation factor, which is a number between 0 and 1. Once a set of process variation parameters are given, we use these parameters as the new mean and scale the range of process variations given in Table 1 by the correlation factor. As a result, higher correlation factors imply less correlation between two random variables. Note that this definition is opposite to that of the correlation coefficient, wherein a higher correlation coefficient implies more correlation between two random variables. The correlation factors are calculated from the vertical and horizontal spatial correlation dependences presented by Friedberg et al. [15]. For each bit in a cache block, we have used a correlation factor of 0.01 and the correlation factor between rows is set to 0.05. Assuming that the ways are laid out on a 2 by 2 mesh, the way that is on the same vertical line with the first way uses the correlation factor 0.45; the way that is on the same horizontal line with the first way uses a the correlation factor 0.375. Finally, the way that is on the same diagonal line with the initial way uses the correlation factor 0.7125.

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

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

4. Yield-Aware Cache Architectures

In this section, we describe our architectural techniques to improve yield. There are two important factors that limit the yield: excessive leakage and excessive delay. To address these two factors, we have developed two types of novel schemes, which are explained in the following sections. First, we discuss a power-down technique that minimizes the leakage consumption and hence increases yield. Then, in Section 4.2, we describe how the naïve power-down technique can be modified to increase the yield even further. This scheme is called the Horizontal Yield-Aware Power-Down (H-YAPD). Section 4.3 discusses a variable access latency cache that aims to minimize yield losses due to delay constrains. In Section 4.4, we describe a hybrid scheme that employs both techniques to boost the yield even further.

4.1. Yield-Aware Power-Down (YAPD)

The Yield-Aware Power-Down (YAPD) technique is based on the Selective Cache Ways (SCW) method [4] combined with the Gated-Vdd technique [30]. Although SCW is implemented for reducing power, we use a similar method for improving yield. Particularly, the YAPD technique disables cache ways based on their delay and power consumptions. If a cache way violates maximum allowed latency constraint, it is turned off. Similarly, if the total power consumption of the cache exceeds the limit, the way with the highest leakage power consumption is turned

off. When a way is disabled, its decoders, pre-charge circuits and the sense-amplifiers are turned off using Gated-Vdd. This practically eliminates both static and dynamic power. Note that once a way is turned off, it will never be used throughout the life of the chip. Due to the fact that YAPD uses the array partitioning that is already present, only minor changes to a conventional cache are required.

Figuring out the ways that need to be disabled can be done during memory testing right after fabrication and/or on the field using leakage power sensors [20]. When we determine that a particular way consistently fails to return data in time, or has excessive leakage, it can be turned off. Figure 4 gives a high-level depiction of the YAPD method implemented for a 4-way cache.

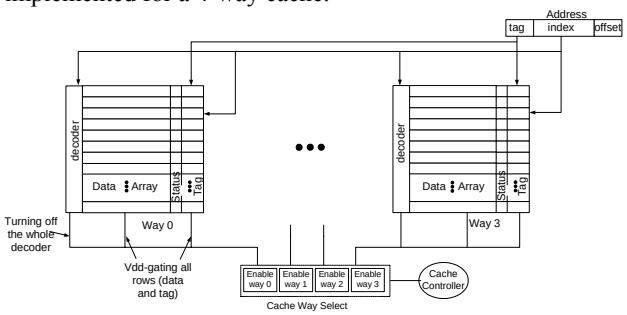

Figure 4. YAPD implementation on a 4-way cache

4.2. Horizontal YAPD (H-YAPD)

One problem with the YAPD scheme is the correlation of process variation parameters between different cache ways. Particularly, since different banks/ways of a cache are implemented physically close to each other, they are strongly correlated. As a result, if one way fails due to delay variation and/or excessive leakage, the remaining ways also fail with high probability. It should be apparent that YAPD scheme described in the previous section would try to close most or all ways. To resolve this limitation, we develop a second yield-aware power-down scheme that disables a subset of rows in all ways. In other words, we effectively turn off a horizontal way, instead of a regular (vertical) way. The reasoning behind this scheme lies in the observation that different paths in a cache show a similar reaction to the same process variation parameters. To understand this behavior, assume that all the cache ways observe the same process variations. Also, assume that the upper-most row in the bank is the critical path, and a second near-critical path is in the middle of the same bank. For a particular set of process variation parameters, the latency of upper-most rows may increase 10%, while the latency of the middle rows will increase by 5%. In another variation, the latency of the middle rows will increase by 10%, while the latency of the upper-most rows will increase by 5%. As a result, for a given process variation, either all the upper-most rows of the ways or all the middle rows will violate the delay constraint. YAPD will then try to turn off all the ways. H-YAPD, on the other hand, will only turn off the sections (e.g., upper-most or middle) of the cache that causes the problem. Clearly, this scenario requires all the cache ways to experience the same or at least similar process variation parameters. Such a behavior is expected, because there is strong spatial correlation: the variation parameters changes slightly from one way to another (c.f., Sections 2 and 3).

One important issue with designing such a cache is that all the sets corresponding to the same address should not be turned off at the same time. To support correct operation, we need to change the decoder structures in each way. Figure 5 shows how we can modify the decoders for each way and which lines are enabled/disabled by each cache way select signal. In this new configuration, all the blocks in a horizontal region corresponds to different addresses. As a result, if we turn a horizontal way off, each address will still have three possible positions. Consider the case where we turn off h-way 0. In that case, a block that is mapped between lines 0 and 31 can reside in vertical ways 1, 2, or 3. Similarly, an address that is mapped to block address 96 through 127 may reside in vertical ways 0, 2, and 3. In every case, we will search the blocks in exactly three locations. As a result, the hit/miss behavior of this architecture will be identical to that of a 3-way cache, i.e., H-YAPD and YAPD will exhibit identical hit/miss behavior. Such a decoder has no area or latency overhead compared to a regular decoder. We only change the configurations of the post-decoders. Figure 6 shows a cache architecture that implements H-YAPD for a 4-way cache where each way has 16 lines.

One disadvantage of the H-YAPD is its increased latency. Since the granularity of the power-down is changed, we see an increase in the average latency of cache accesses. Simulations on the HSPICE model show a 2.5% increase in the access latencies on average. In addition, since some parts of the decoder as well as pre-charge and sense amplifier circuits cannot be turned off completely, the power savings may be lower than the YAPD.

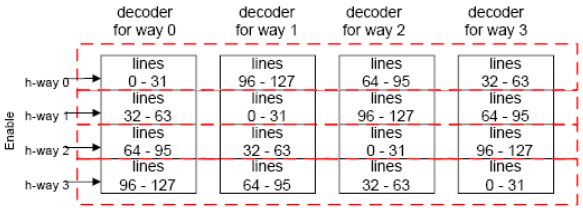

Figure 5. The high-level view of the decoders in the H-YAPD implementation

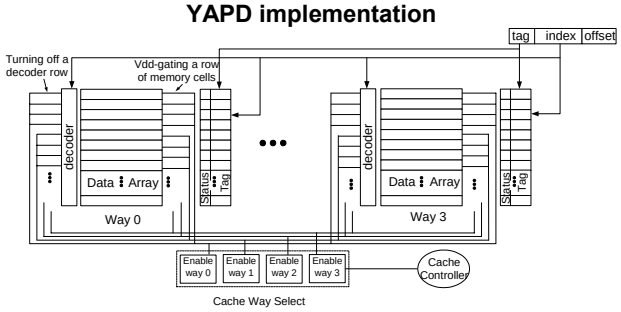

Figure 6. H-YAPD implementation on a 4-way cache

The disadvantage of both YAPD and H-YAPD is the possibility of increased cache miss rates and hence increase in application execution time. Although we aim to minimize

yield loss, we cannot allow unbounded performance degradation. *Therefore, we set a limit of 2% average performance degradation. Our results with the SPEC2000 applications reveal that we can turn off a single way in our 16 KB, 4-way set-associative cache within this budget.*

4.3. Variable-Latency Cache Architecture

A major problem with the power-down schemes described above is that they may have high performance degradations due to increased cache miss rates. An alternative method is to keep the slower ways enabled, but allow them to complete after additional cycles. This effectively results in a cache architecture with variable access latency capability.

The idea of variable access latency caches is not new. Kim et al. proposed the NUCA [19] architecture to manage large on-chip caches. By exploiting the variation in access time across subarrays, NUCA allows fast access to close sub-arrays while retaining slow access to far subarrays. Intel's Montecito [28] uses a similar approach where each core has a shorter access time to its local L2 cache slices and a longer delays to remaining L2 cache slices. However, previous studies focused on implementing this idea on higher level caches. In addition, to the best of our knowledge, no existing work employs the variable access latency for yield optimization.

An important issue with implementing a variable latency level 1 cache is the capability of forwarding the values that are read by the load instruction to the dependent instructions. Without a variable latency load operation, the time that the loaded data is available is known exactly (assuming a load hit). However, with the variable latency cache, the data will be available in a window of cycles. Therefore, we need to augment the corresponding structures in the datapath such that when the data is available from the cache, it is sent to the correct destination.

After a load instruction is scheduled, the dependent instructions start executing assuming the load access will take shortest possible amount of time, i.e., cache hit latency, which is four cycles in our architecture. To avoid these instructions from reaching the execute stage before the data from the cache is available, we add buffers at the inputs of each functional unit. These buffers are called *load-bypass buffers*. Each buffer has entries that will allow the instruction to wait until the data is available. These entries will be used if a load does not complete in four cycles. In our architecture, the load-bypass buffers have a single entry to allow accesses of 4 or 5 cycles. It is possible to add more buffers to support a larger variation (e.g., ways with 6 or 7 cycle access latencies), however, our results revealed that the additional yield optimizations with this extension are minor and the performance degradation can be very high. Therefore, we have decided to allow only one additional cycle in cache accesses. Once the cache access is complete, the destination register number and the data read from the cache are forwarded to the load-bypass buffers; where each entry compares the stored register number (which is the input register for the dependent instruction) with the forwarded value. If the two values are identical, the data (i.e., the output of the load operation) is latched into the buffer. Then, in the next cycle the operation will start execution. Figure 7 shows the hardware for this approach. Note that we omitted the multiplexers at the inputs of the functional unit that selects from different forward values. If the input operands of an instruction are ready, i.e., no forwarding is needed from the cache; the operation can simply skip the buffer and start the execution. However, if one of the input operands will be provided by a delayed load operation, the instruction will enter the buffer. Once it receives the data, it will move to the function unit as described above. If there is another instruction depending on this stalled operation, it also has to be stalled. This chain may continue for more operations depending on the latency of operations and the number of pipeline stages between scheduling and execution. To illustrate this, consider the instructions L1, D1, D2, …, Dn, where D1 is dependent on L1 and D2 is dependent on D1, etc. Once the load is delayed, D1 is stalled for a cycle in a load-bypass buffer. At the same time, the scheduler is informed about this stall; hence the scheduling of any direct or indirect dependent instructions are delayed for one cycle. However, depending on the time between scheduling decision and the start of the execution, several operations may already been scheduled to execute. These operations will have to use the load-bypass buffers to stall for an additional cycle. In such cases, these dependent operations will receive their data from a function unit rather than the data cache. Therefore, the load-bypass buffers have to be connected to not only the data cache, but also to all the function units.

If an instruction is in the load-bypass buffer but does not receive its input, it means that the load access missed in the cache. Therefore, the dependent instruction needs to be flushed and re-executed based on the replay mechanism that is employed in the processor. Note that the complexity of replay and the miss penalty of the load operations are not affected by our variable latency architecture.

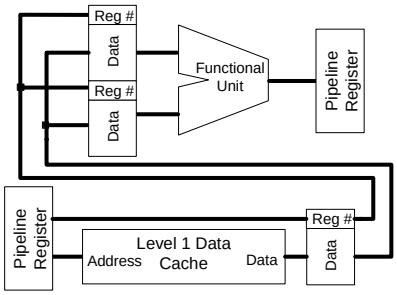

Figure 7. Implementation of the load-bypass buffer and the associated forwarding from the data cache

To implement VACA, the scheduler in the processor has to be modified. Each instruction that is dependent on a load should be marked such that it will enter the buffer rather than going to the functional units if the load is delayed. In addition, the scheduler needs to stall an instruction if it is indirectly dependent on a delayed load.

4.4. Hybrid Scheme

YAPD and VACA mainly target different causes of yield loss. Therefore, we have devised a hybrid method that aims to combine the positive aspects of YAPD and VACA architectures. In this scheme, we propose using a hybrid cache that implements a YAPD scheme to turn off cache blocks in combination with the VACA. This idea can be applied to H-YAPD as well.

Similar to the VACA architecture, we implement a cache capable of allowing 4 and 5 cycles of access latency. If there are ways that need more than 5 cycles or violate the leakage power limitations, they can be disabled with the power-down mechanism. One advantage with the Hybrid mechanism is that it has many options to implement. For example, if two of the ways require 4 cycles and the other two require 5 cycles of latency, the hybrid scheme can choose to keep both 5-cycle ways enabled and work as the VACA architecture or it can disable them and work as YAPD (or H-YAPD) scheme or it can choose to disable one 5-cycle way and have a performance level between the two schemes. This choice depends on the behavior of the executed application. If the application is a memory intensive one, disabling a way would hurt the performance more than keeping it enabled and accessing it with 5 cycles. On the contrary, for a computation intensive application the overhead for turning off a single way may be less than accessing that way using 5 cycles. Although there are several possibilities, in this work, we keep a fixed hybrid architecture. Our Hybrid scheme chooses to keep the ways on as long as possible. In other words, Hybrid scheme turns off a way only if its delay exceeds 5 cycles or it has excessive leakage power consumption causing a violation. In addition, similar to YAPD and H-YAPD, Hybrid scheme can turn off at most a single way. Caches that cannot be saved under this constraint still cause yield loss.

4.5. Naïve Alternatives

The easiest way to utilize the chips that violate latency limitations is to group the fabricated chips into separate bins with different performance levels. For instance, if we detect that a particular way in the cache requires an extra cycle to fetch the data, while all the other ways operate normally with 4 cycles of access latency, we could set the instruction scheduling logic such that it always expects the result to be ready in 5 cycles. Architectural simulations performed using SimpleScalar showed that the approach has high performance overheads: 6.42% on average for the SPEC2000 applications if the accesses require an additional cycle and 12.62% if they require two additional cycles for correct operation.

5. Experimental Results

To analyze the effectiveness of the proposed schemes, we need to analyze two different aspects: their impact on the yield and their impact on performance. In the next section, we first describe the results analyzing their impact on yield. Section 5.2 discusses the performance implications. *We must note that the proposed schemes are only activated when a chip does not meet design criteria, i.e., when a chip would otherwise be discarded due to parametric yield loss. Therefore, the schemes do not have any performance impact on the rest of the chips that pass the testing.*

5.1. Yield Results

In the core of the analysis lies our capability to estimate yield loss of a particular design. Therefore, we first present the methodology used to estimate yield loss. Similar to the methodology by Rao et al. [32], we first model 2000 different caches under process variations. To achieve this, we perform HSPICE simulations for 2000 caches, where each simulation picks a different set of process variation parameters from their respective intervals discussed in Section 3. Note that as we have described in Section 3, each simulation models 4 different cache ways by considering different critical/near-critical paths for that particular way. After each simulation, we compare the address-to-data output latency of each path in a way and the maximum of these numbers gives the access latency for that cache way. Similarly, the maximum among all way latencies becomes the cache access latency. Using the same simulations, we also find the total leakage power consumption of each cache by summing over the leakage power consumed by each way. Figure 8 shows the distribution of normalized leakage power consumption versus distribution of cache access latencies.

Once the latency versus leakage distribution is found, the yield can be calculated by setting power and performance limits. Rao et al. [32] analyze an ALU and use mean+sigma value for performance limit and 1.75xaverage leakage power limit for a 65 nm technology. Similar to their approach, we use the same performance limit. However, we picked the power limit at 3 times the average leakage power to compensate for the increased variation in 45 nm and the different component we are studying (cache instead of ALU). Table 2 tabulates the distribution of the sources of parametric yield loss encountered in the base case and when we implement the YAPD, VACA, and Hybrid schemes. The total number of chips is 2000. Hence, with this architecture, the expected parametric yield loss is 16.9%. Similarly, Table 3 shows the results when we implement H-YAPD.

As highlighted in Section 4.2, the cache architecture used in the H-YAPD scheme is slightly different than the YAPD architecture. Therefore, we have performed a different set of HSPICE simulations for 2000 caches representing the new architecture. We have applied the same process variation parameters used in the previous simulations. On average, we see that the delay of the architecture increases by 2.5%. As a result, the number of chips that do not meet the delay constraint increases. Particularly, for the base case of this cache architecture, we see that 18.1% of the chips are lost.

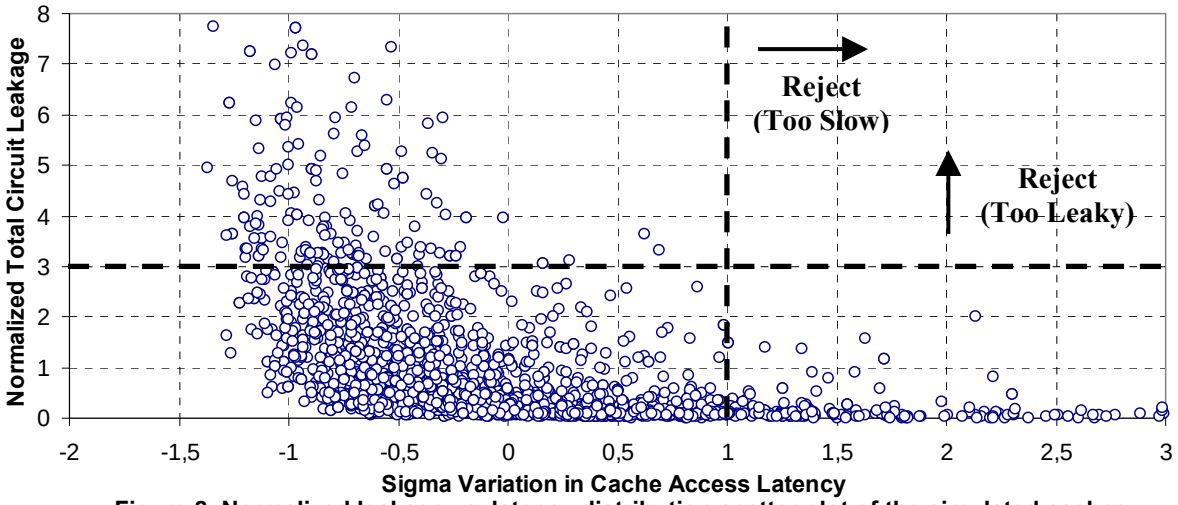

Figure 8. Normalized leakage vs. latency distribution scatter plot of the simulated caches

Table 2. Sources of yield loss for regular power-down

Reason of Loss	# Chips	Losses with Scheme		
		YAPD	VACA	Hybrid
Leakage Constraint	138	33	138	33
Delay Constraint (1 Way)	126	0	34	0
Delay Constraint (2 Ways)	36	36	20	7
Delay Constraint (3 Ways)	23	23	19	11
Delay Constraint (4 Ways)	16	16	15	13
Total	339	108	226	64

Table 3. Sources of yield loss for horizontal power-down

Reason of Loss	# Chips	Losses with Scheme		
		H-YAPD	VACA	Hybrid
Leakage Constraint	138	26	138	26
Delay Constraint (1 Way)	142	0	38	0
Delay Constraint (2 Ways)	33	33	17	6
Delay Constraint (3 Ways)	29	24	21	12
Delay Constraint (4 Ways)	20	17	19	16
Total	362	100	233	60

Table 4. Total yield losses for relaxed and strict cases with regular power-down

	# Chips	Losses with Scheme		
		YAPD	VACA	Hybrid
Relaxed	184	51	124	25
Strict	727	234	503	144

Table 5. Total yield losses for relaxed and strict cases with horizontal power-down

	# Chips	Losses with Scheme		
		H-YAPD	VACA	Hybrid
Relaxed	191	51	131	25
Strict	752	224	516	146

We observe that YAPD and H-YAPD schemes decrease the parametric yield loss by 68.1% and 72.4%, respectively. Considering the overall yield, YAPD and H-YAPD increases the total yield to 94.6% and 95.0%, respectively. YAPD and H-YAPD reduces the losses due to leakage by 76.1% and 81.2%, respectively. They also nullify the losses due to a single way delay violations. VACA has a lower decrease in yield loss. As discussed in Section 4.3, all the chips with 6 or more cycles of cache access latency are considered as yield loss with VACA. Furthermore, VACA does not improve yield losses due to leakage. Overall, VACA reduces the yield loss by 33.3% for regular power-down and 35.6% for horizontal power-down caches. As expected, the Hybrid scheme performs the best in terms of yield improvement. Combining the benefits of both schemes, the Hybrid scheme reduces the losses due to leakage, eliminates all single way delay violation losses, and achieves further reduction in losses for multiple-way delay violations. The Hybrid scheme results in 81.1% and 83.4% reduction in parametric yield loss, for regular and horizontal power-down caches, respectively. Using the Hybrid scheme, the yield improves to 96.8% and 97.0% for the regular and horizontal power-down caches, respectively.

An important aspect of yield loss modeling is the ability to change the constraints. Depending on the architecture, manufacturers can change the constraints on their chips. This also could be modeled with our approach. Therefore, we analyzed the advantages of our schemes while changing the yield requirements. Tables 4 and 5 show the cumulated yield loss numbers for the 2000 simulation points. For the relaxed constraint, we set the power limit at 4 times the average

leakage power and the performance limit at mean+1.5xsigma. Similarly, the strict case uses 2 times the average leakage power for the power limit and mean+0.5xsigma as the performance limit. The results show that the proposed schemes perform fairly under different yield constraints. Particularly, the Hybrid scheme achieves 98.8% yield for the relaxed constraint and approximately 92.8% yield for the strict constraint.

5.2. Performance Implications

The yield results for the proposed schemes were explained in the previous section. It is important to note that for the simulation points that did not need any yield-aware technique to perform correctly, our proposed schemes have no performance overhead. However, for the cases that are converted from yield loss to gain, each scheme has a different performance overhead depending on the configuration of the cache. For example, a cache with three ways of 4 cycle and one way of 5 cycle access latency requirement will exhibit a different behavior than a cache with a two 4-cycle and two 5-cycle ways. To understand and compare the effectiveness and limitations of each scheme we must further look into the performance implications caused by them for different cache configurations.

Table 6. Performance degradation of SPEC2000 applications for different cache configurations using the YAPD, VACA, and Hybrid schemes.

Number of ways with delay			Chip frequency	Performance degradation [%]		
4	5	6+		YAPD	VACA	Hybrid
3	**1**	**0**	91	1.08	1.81	1.81
2	**2**	**0**	16	N/A	3.32	3.32
1	**3**	**0**	4	N/A	5.47	5.47
0	**4**	**0**	1	N/A	6.42	6.42
3	**0**	**1**	35	1.08	N/A	1.08
2	**1**	**1**	13	N/A	N/A	3.65
1	**2**	**1**	8	N/A	N/A	5.49
0	**3**	**1**	2	N/A	N/A	7.39
4	**0**	**0**	105	1.08	N/A	1.08
Weighted Sum				1.08	2.20	1.83

The SimpleScalar 3.0 [35] simulator is used to measure the effects of the proposed techniques on performance. The necessary modifications to the simulator have been implemented to perform selective replay, the scheduler, the busses between caches, and port contention on caches. We have also made changes to SimpleScalar to simulate a realistically sized issue queue, and to model the events in the issue queue in detail. We simulate 13 floating-point and 11 integer benchmarks from the SPEC2000 benchmarking suite [37]. We simulate 100 Million instructions after fast-forwarding application-specific number of instructions as proposed by Sherwood et al. [34].

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 separate level 1 instruction and data caches: level 1 instruction cache is 16 KB, 4-way associative with 64-byte block size and 2 cycle latency, and the level 1 data cache is a 16 KB, 4-way associative with 32-byte block size and 4 cycle latency. Unified level 2 cache is 512 KB, 8-way associative cache with 128-byte block size and 25 cycle latency. The memory access delay is set to 350 cycles. All caches are lock-up free. The simulated architecture implements 7 pipeline stages between the schedule and execute stages.

Table 6 lists the different cache way latency configurations encountered during HSPICE simulations that are converted from yield loss to yield gain along with the number of times they are encountered. In addition, it lists the average performance degradation for SPEC2000 applications caused by each of the schemes for a given configuration. The configuration 3-1-0 corresponds to the case where three out of four ways in a cache has 4-cycle access latency, whereas the fourth way requires an additional access cycle for correct operation. A configuration 3-0-1, on the other hand, corresponds to a cache with three ways requiring 4 cycles and another one requiring 6 or more cycles. *Note that the performance degradations of the YAPD and H-YAPD schemes are identical for a given cache configuration.* If multiple ways require 5 or more cycles, the chip is lost with the YAPD schemes. Similarly, the number of ways requiring 6 or more cycles of access should be equal to zero for VACA and at most one for Hybrid schemes. The last configuration, 4-0-0 shows the leakage power limited caches that did not violate the timing requirements. In this case, we need to shut down a way to reduce the leakage power consumption. Since VACA does not disable ways, it cannot save any cache with this configuration. The results presented in the bottom row of Table 6 correspond to a weighted sum of the performance degradation on the chips that are saved, i.e., the performance degradation for a configuration is multiplied by its fraction within the saved chips and summed over all configurations. To illustrate this, consider the sum for VACA. This number is achieved by first finding the total number of chips saved by it (112). Then, the fraction of each saved configuration is found (e.g., 91/112=0.81 for 3-1-0). We multiply this fraction with the corresponding performance degradation (e.g., 0.81*1.81%=1.46%) and sum over all configurations (1.46%+0.47%+0.20%+0.06%=2.20%). This number corresponds to average performance degradation for the batch of chips that are saved. On average, for the saved chips, YAPD, VACA, and Hybrid schemes cause 1.1%, 2.2%, and 1.8% increase in the CPI, respectively.

Figures 9 and 10 show the increase in CPI for different SPEC2000 applications for two frequent cache configurations listed above: 3-1-0 and 2-2-0, respectively. In the case of 3-1-0, the hybrid scheme can either choose to close the 5-cycle latency way in this case like YAPD or keep it enabled where its performance would be equivalent to VACA scheme. However, as we have described in Section

4.4, a way is turned off only if necessary. Therefore, Hybrid scheme behaves identical to the VACA. Note that, YAPD can turn off at most a single way. Therefore, it exhibits the same performance degradation for each cache configurations it saves (3-1-0, 3-0-1 and 4-0-0). Similarly, YAPD and Hybrid schemes would have equal performance results for 3-0-1: each will close a single way and access the remaining three ways in 4 cycles. For the 3-1-0 configuration, the CPI is increased by 1.1% and 1.8% on average for the YAPD and VACA (and hence Hybrid) schemes, respectively. Figure 10 shows the performance degradation for SPEC2000 applications when the cache configuration is 2-2-0. Since there are two ways that have 5 cycle access latency, YAPD schemes do not operate correctly. The performance degradation for both VACA and Hybrid schemes is 3.3% on average. The average increases in CPI for other cache configurations are presented in Table 6.

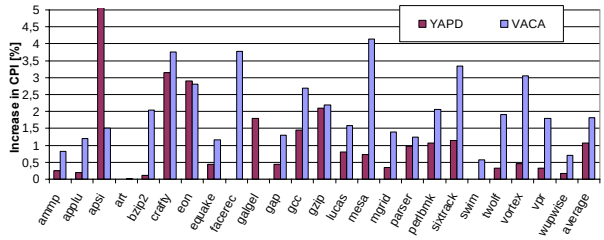

Figure 9.Increase in CPI for YAPD and VACA for cache configuration 3-1-0

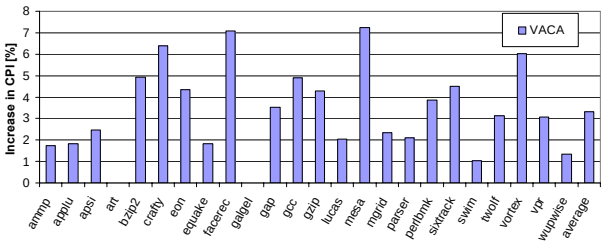

Figure 10.Increase in CPI for VACA for cache configuration 2-2-0

6. Related Work

Due to the effects of process variations, yield improvement has been an important problem for circuit designers as well as manufacturers since early 1970s. Recently, yield losses have become a major concern due to aggressive technology scaling. A number of circuit level techniques have been proposed to counter the effects of process variations [6, 9, 10, 31, 33, 36]. Most of these techniques focus on analyzing the design, either statistically or by using static timing analysis, and then modifying the parts that are more susceptible to variation, e.g., sizing the gates on the critical and near-critical paths which are likely to violate timing requirements [9, 31, 36]. Although these methods help increase the yield, their savings are limited since they cannot save any of the chips that violate the limitations after fabrication. Also, with increasing process variation effects due to smaller technologies, the number of candidate elements for sizing increases, complicating the application of such enhancements.

Performance binning is another long-standing approach to increase yields [6, 10, 33]. Circuits are placed in separate bins depending on their performance and power consumption levels and marketed with different prices for each bin. As a result, lower performing bins can be sold at a lower price instead of merely getting thrown away. In this regard, Borkar et al. [6] combine the circuit level approaches with frequency binning. They propose a number of circuit level techniques to control body bias voltage, supply voltage, and temperature to get higher frequency bins.

There are a number of architectural and system level approaches that can impact yield. Datta et al. [11] employs gate sizing to optimize individual stages in the processor pipeline optimizing yield for a given area constraint or minimizing area for a given yield constraint using the concept of area borrowing. Kurdahi et al. [21] demonstrates analysis techniques to model and improve the yield of SRAMs at the system level by proper accounting for the coupling between the algorithms targeted for an SoC and the performance, power, and yield of SRAMs used in implementing them. These studies attack the yield loss problem at a lower level compared to our methods. Agarwal et al. [3], on the other hand, propose a new cache design where individual cache lines can be turned off if they are found to be faulty. This approach resembles our yield-aware power-down schemes. However, their work concentrates on direct mapped caches. Also, their fine-grained approach neglects the spatial correlation among different circuit elements, making decoder and the cache controller unnecessarily complicated. Another work by Datta et al. [12] tries to predict the yield of a pipelined circuit with analytical models. They observe that introducing imbalances among paths in a pipeline stage actually increases the yield of the design. Finally, several approaches, such as Razor by Ernst et al. [14], can be applied to improve yield. However, to the best of our knowledge, the yield impact of such schemes has not been studied.

7. Conclusions

Reducing chip yields due to process variations is an important problem for circuit designers as well as manufacturers. In this work, we propose four microarchitectural techniques to minimize the yield loss due to power and delay violations in the data cache. We first discuss the major sources of process variations and model them using HSPICE. Then, we introduce our schemes. The first scheme, Yield-Aware Power-Down (YAPD), disables a cache way if it violates the delay or power limits. The second scheme, Horizontal-YAPD (H-YAPD), modifies the approach by turning off horizontal regions of a cache instead of the regular (vertical) ways. Due to spatial correlation of process variations, turning off segments that exhibit similar behavior improves the yield further. The third scheme, VAriable-latency Cache Architecture (VACA), allows different load accesses to be completed with varying

latencies with the help of special buffers placed before the function units. As a result, if some accesses take longer than the predefined number of cycles, the execution can still be performed correctly. A final method, Hybrid scheme, combines YAPD and VACA. Experimental results demonstrate that the yield losses can be reduced by 68.1% and 72.4% with YAPD and H-YAPD schemes and by 33.3% and 81.1% with VACA and Hybrid mechanisms, respectively, improving the overall yield to as much as 97.0%.

8. References

[1] "EDA Tools Aim at Improving Yield," http://www.eurosemi.eu.com/front-end/features-full.php?id=6762

[2] Agarwal, A., D. Blaauw, and V. Zolotov. "Statistical Timing Analysis for Intra-die Process Variations with Spatial Correlations," in Proc. Intl. Conf. on Computer-Aided Design. 2003. San Jose, CA.

[3] Agarwal, A., et al., "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies," IEEE Trans. Very Large Scale Integrated Systems, 2005. 13(1): p. 27-38.

[4] Albonesi, D. "Selective Cache Ways: On-demand Cache Resource Allocation," in Intl. Symposium on Microarchitecture. Nov. 1999. Haifa, Israel.

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

[6] Borkar, S., et al. "Parameter Variations and Impact on Circuits and Microarchitectures," in Proc. of the Design Automation Conference. 2003. Anaheim, CA.

[7] Cao, Y., et al. "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," in Custom Integrated Circuits Conference. 2000. Orlando, FL. http://www.eas.asu.edu/~ptm

[8] Chang, H. and S.S. Sapatnekar. "Statistical Timing Analysis Considering Spatial Correlations Using a Single PERT-like Traversal," in Proc. Intl. Conf. on Computer-Aided Design. 2003. San Jose, CA.

[9] Choi, S.H., 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. 2004: San Diego, CA. p. 454-459.

[10] Datta, A., et al. "Speed Binning Aware Design Methodology to Improve Profit Under Parameter Variations," in Proc. of the Conf. on Asia South Pacific Design Automation. 2006. Yokohama, Japan.

[11] Datta, A., et al., "Delay Modeling and Statistical Design of Pipelined Circuit Under Process Variation," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2006.

[12] Datta, A., et al. "Yield Prediction of High Performance Pipelined Circuit with Respect to Delay Failures in Sub-100nm Technology," in IEEE Intl. On-Line Testing Symposium. 2005. France.

[13] Duvall, S.G. "Statistical Circuit Modeling and Optimization," in Intl. Workshop on Statistical Metrology. 2000.

[14] Ernst, D., et al. "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in Proc. of the Intl. Symposium on Microarchitecture. 2003. San Diego, CA.

[15] Friedberg, P., et al. "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization," in Proc. of the Intl. Symposium on Quality of Electronic Design. 2005. San Jose, CA.

[16] Goering, R. "90-, 65-nm Yields Prey to Leakage," http://vlsicad.ucsd.edu/leakage.htm

[17] Humenay E., D. Tarjan, K. Skadron. "Impact of Parameter Variations on Multi-Core Chips," in the Workshop on Architectural Support for Gigascale Integration. 2006. Boston, MA.

[18] Jones, H.H. "A Delayed 90-nm Surprise," http://www.designchain.com/column.asp?id=2&issue=fall04

[19] Kim, C., D. Burger, and S.W. Keckler. "An Adaptive, Non-uniform Cache Structure for Wire-delay Dominated On-chip Caches," in Proc. of the Intl. Conf. on Architectural Support for Programming Languages and Operating Systems. 2002. San Jose, CA.

[20] Kim, C.H., et al. "An On-Die CMOS Leakage Current Sensor For Measuring Process Variation in Sub-90nm Generations," in VLSI Circuits Symposium. 2004. Honolulu, HI.

[21] Kurdahi, F.J., et al. "System-Level SRAM Yield Enhancement," in International Symposium on Quality Electronic Design, Mar. 2006. San Jose, CA.

[22] Mani, M., A. Devgan, and M. Orshanksky. "An Efficient Algorithm for Statistical Minimization of Total Power Under Timing Yield Constraints," in Proc. of the Design Automation Conference, 2005, Anaheim, CA.

[23] Mehrotra, V. "Modeling the Effects of Systematic Process Variation on Circuit Performance," PhD Thesis, Dept. of EECS, MIT, 2001.

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

[25] Miller, M. "Nanometer Yield Enhancement Begins In The Design Phase," http://www.elecdesign.com/Articles/Index.cfm?AD=1&AD=1&ArticleID=9494

[26] Mudge, T. "Low Power System's on a Chip - Today's Challenge," in Intl. Seminar on Application-Specific Multi-Processor SoC. Jul. 2004. http://tima.imag.fr/mpsoc/2004/slides/Mudge.pdf

[27] Mui, M.L., K. Banerjee, and A. Mehrotra. "Power Supply Optimization in Sub-130 nm Leakage Dominant Technologies," in Proc. of the Intl. Symposium on Quality Electronic Design. 2004. San Jose, CA.

[28] Naffziger, S., et al., "The Implementation of a 2-core, Multi-Threaded Itanium Family Processor," IEEE Journal of Solid-State Circuits, 2006. 41(1): p. 197-209.

[29] Nassif, S.R. "Modeling and Analysis of Manufacturing Variations," in IEEE Conference on Custom Integrated Circuits. May 2001. San Diego, CA.

[30] Powell, M., et al. "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in ACM/IEEE Intl. Symposium on Low Power Electronics and Design. 2000. Rapallo, Italy.

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

[32] Rao, R.R., et al., "Modeling and Analysis of Parametric Yield under Power and Performance Constraints," IEEE Des. Test, 2005. 22(4): p. 376-385.

[33] Raychowdhury, A., et al. "A Novel On-chip Delay Measurement Hardware for Efficient Speed Binning," in Intl. Online Testing Symposium. Jul. 2005. France.

[34] Sherwood, T., E. Perelman, and B. Calder. "Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications," in Proc. of the Intl. Conf. on Parallel Architectures and Compilation Techniques. 2001.

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

[36] Sinha, D., N.V. Shenoy, and H. Zhou. "Statistical Gate Sizing for Timing Yield Optimization," in Proc. Intl. Conf. on Computer-Aided Design. 2005. San Jose, CA.

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

[38] Visweswariah, C., et al. "First-Order Incremental Block-Based Satistical Tming Aalysis," in Proc. of the Design Automation Conference. 2004. San Diego, CA.

[39] Wilson, D. "NVIDIA's Tiny 90nm G71 and G73: GeForce 7900 and 7600 Debut," http://www.anandtech.com/video/showdoc.aspx?i=2717