Process Variation Tolerant 3T1D-Based Cache Architectures

Xiaoyao Liang, Ramon Canal∗, Gu-Yeon Wei and David Brooks
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138
∗Dept. of Computer Architecture, Universitat Politecnica de Catalunya, Barcelona, Spain
{xliang,guyeon,dbrooks}@eecs.harvard.edu, ∗rcanal@ac.upc.edu

Abstract

Process variations will greatly impact the stability, leakage power consumption, and performance of future microprocessors. These variations are especially detrimental to 6T SRAM (6-transistor static memory) structures and will become critical with continued technology scaling. In this paper, we propose new on-chip memory architectures based on novel 3T1D DRAM (3-transistor, 1-diode dynamic memory) cells. We provide a detailed comparison between 6T and 3T1D designs in the context of a L1 data cache. The effects of physical device variation on a 3T1D cache can be lumped into variation of data retention times. This paper proposes a range of cache refresh and placement schemes that are sensitive to retention time, and we show that most of the retention time variations can be masked by the microarchitecture when using these schemes. We have performed detailed circuit and architectural simulations assuming different degrees of variability in advanced technology nodes, and we show that the resulting memory architecture can tolerate large process variations with little or even no impact on performance when compared to ideal 6T SRAM designs. Furthermore, these designs are robust to memory cell stability issues and can achieve large power savings. These advantages make the new memory architectures a promising choice for on-chip variation-tolerant cache structures required for next generation microprocessors.

1. Introduction

Nanoscale technology scaling offers the promise of continuing transistor density and performance trends. However, the road is fraught with difficulties resulting from increased process variations that limit performance gains and affect stability of key circuit blocks such as on-chip memories. Addressing these problems will require innovation from all levels in the design flow from the devices to the system architecture. This paper investigates using dynamic memory cells with architecture support to enable robust cache designs tolerant to process variations for future generations of high-performance microprocessors.

Process variation is mainly caused by fluctuations in dopant concentrations and device channel dimensions. Gate length variation can change the effective driving capability of the transistor, as well as the threshold voltage due to the short channel effect. Random dopant variations can also change the threshold voltage of the device. The nature of the semiconductor manufacturing process gives rise to both within-die variations (i.e. device features on one chip can be different) and die-to-die variations (i.e. device features across chips can be different). As technology scales, within-die variations are getting larger, significantly affecting performance and compromising circuit reliability.

On-chip memories consume a significant portion of the overall die space in modern microprocessors given slow off-chip memories and due to the area efficiency and high system performance they offer in exchange for the space and power they consume. On-chip caches rely on SRAM cells, which have generally scaled well with technology. Unfortunately, stability, performance, and leakage power will become major hurdles for future SRAMs implemented in aggressive nanoscale technologies due to increasing device-to-device mismatch and variations. Circuit imbalance due to mismatch can compromise cell stability and exacerbate leakage for traditional SRAM designs. Furthermore, the large number of critical paths and the small number of devices in each path both contribute to increase access time variability. This variability is especially important for higher levels of the memory hierarchy that are more latency critical. In contrast, while lower levels of the memory (e.g., L2 cache) are less sensitive to latency, they remain susceptible to bit flips. One simple solution to these problems is to slow down scaling of SRAMs at the expense of lower performance and larger area. However, this would effectively mean the end of Moore's Law scaling of transistor density and speed for future processor designs.

To avoid these scaling limitations, new circuit and architectural solutions are needed. In this paper, we investigate on-chip memory architectures based on 3T1D dynamic memory cells [18]. We demonstrate a robust memory design in the context of an important latency-critical component of the processor—the L1 data cache—identified by several researchers as one of the most susceptible architectural components to process variations [2, 11, 22]. We show how characteristics of the modern processor architecture (e.g., superscalar, out-of-order execution) can mitigate the costs associated with dynamic memories and overcome effects of physical device variations, while offering significant performance, stability, and power advantages. The proposed design makes it easier to apply fine-grained variation control schemes compared to traditional SRAM designs. Such fine-grained control is very important for designs that suffer large within-die process variations. The paper discusses various data retention schemes—different refresh and replacement policies—and provides detailed comparisons in performance and power. This paper takes several steps in the direction of variation-tolerant memory architecture designs. Specifically, there are three major contributions:

- Given the transient nature of data flow, dynamic memories are good candidates for data storage structures within the processor core. We propose to use a 3T1D-based dynamic memory cell as the base for a new generation of on-chip memory designs.
- The proposed cache structure can tolerate very large process variation with small or even no impact on performance by intelligently choosing the retention schemes. Effects of process variations can be absorbed into a single parameter – data retention time – efficiently addressed by architectural solutions.
- Besides performance benefits, the proposed caches are robust to memory cell stability issues and can achieve large power savings.

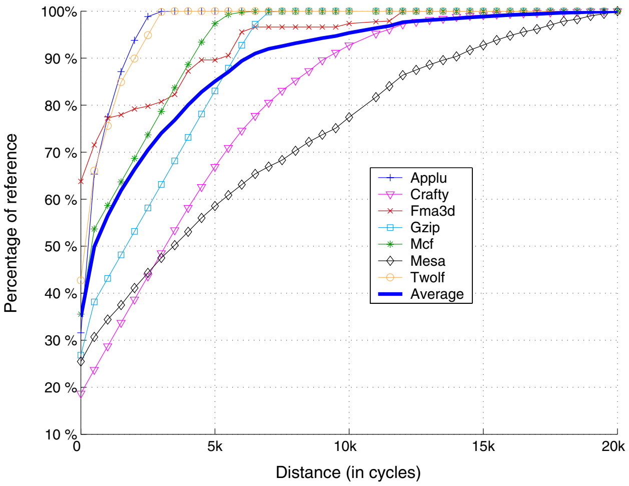


Figure 1. Percentage of cache reference with the elapsed clock cycles.

In the following section, we investigate how process variations affect traditional 6T SRAM designs and provide a detailed comparison to 3T1D DRAMs. Section 3 describes the circuit and architecture simulation methodology used for the analysis presented in Section 4, which shows how dynamic memories can lead to cache designs tolerant to process variations. In order to thoroughly understand the impact of variations, Section 5 presents a sensitivity analysis with respect to different retention time scenarios. We discuss background information and related work in Section 6. Finally, this work is summarized in Section 7.

2. Comparison Between 6T Static Memory and 3T1D Dynamic Memory

For decades, the 6T-based SRAM has been the de facto choice for on-chip memory structures within the processor core such as register files and caches. DRAM memory technologies have been used for larger off-chip caches or main memories. These choices have been driven primarily by the high memory access speed provided by SRAMs and the high density provided by DRAMs. Because the memory storage node in DRAMs is a capacitor, data is stored temporarily and the memory requires periodic refresh if the data needs to be held for extended time periods.

On the other hand, a large fraction of the data consumed in the processor core is transient [6, 31]. As shown in Figure 1, most cache accesses happen within the initial 6K clock cycles after the data is loaded. As long as the data retention time of dynamic memory can meet the system requirements, it is actually a better choice for on-chip memory structures that provide temporary data storage, because the temporal characteristics of the data in the processor may reduce or even eliminate the need for refresh. In light of 6T SRAM scaling concerns, this paper investigates the necessary architectural support that can enable the use of dynamic memories for on-chip L1 data caches.

2.1 Limitations of 6T SRAM Under Process Variations

Process variations can affect the access speed of an SRAM array. A typical 6T SRAM cell is illustrated in Figure 2(a) and the read and write operations are shown in Figure 2(b). In order to read the stored data, one side of the cell typically discharges one of two precharged bitlines. Any variations in the gate length or threshold voltage in the cell's transistors (*T1* or *T2*) can vary the current driving capability of the read path. Under within-die process variation, each memory cell and memory line can have different speeds. Due to design complexity limitations, it may be difficult to to apply fine-grained access approaches to 6T SRAMs, e.g., accessing different lines with different frequencies. In other words, one worst-case cell in the SRAM array can overly burden design margins and timing windows or negatively affect the speed and/or power of the entire structure. Given the criticality of L1 cache latency, the degraded speed of the SRAM directly translates to performance loss for the entire microprocessor or requires overdesign of the cache to cover worst-case corners.

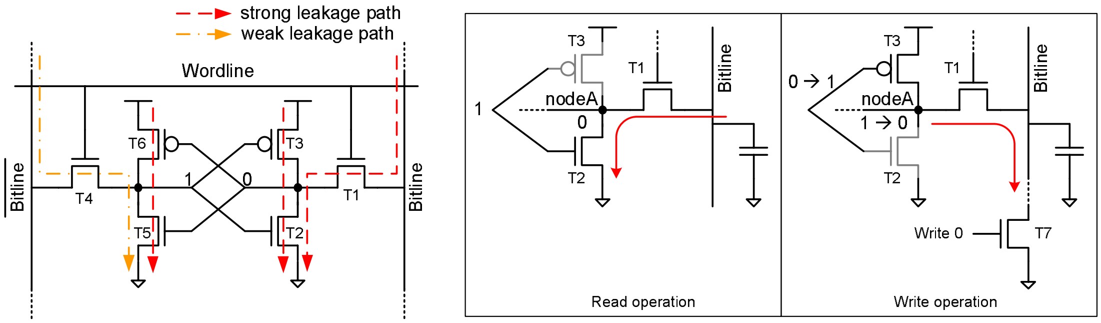
In addition to the performance scalability concerns discussed above, 6T SRAM cell stability will be a major hurdle for future VLSI designs due to device-to-device mismatch. For reliable read operations, transistors *T1* and *T2* in Figure 2b must be carefully sized in order to prevent *nodeA* from rising and inadvertently flipping the bit (i.e. pseudo-destructive read). For writes, the combined effective resistance of the access transistor *T1* and write transistor *T7* must be low enough to overcome the pull-up strength of *T3* and flip the bit. Hence, the read and write operations require a delicate balance of device drive strengths to ensure proper operation. Unfortunately, increasing device mismatch leads to imbalances in the 6T cell that can compromise its ability to read and write reliably. Studies show that even a low error rate in the SRAM can cause large performance loss [2]. Our circuit-level SRAM simulations reveal bit-level flip rates on the order of 0.4% at the 32nm technology node, similar to [2]. Although this rate appears small, it may have large ramifications. For example, in a data cache, line-level redundancy is straightforward to implement, but is ineffective because 256-bit lines would experience a 64% probability of line failure (i.e., 1-0.996^{256}), which is not acceptable.

Leakage power is another major concern for SRAMs. It is well known that leakage power is poised to become the dominant source of power consumption in future digital chips. As shown in Figure 2(a), there are three strong-leakage paths in one 6T SRAM cell since only one transistor is "off" along each path. There are additional weak-leakage paths with stacked transistors along their paths. Such an SRAM structure consumes considerable amounts of static current in order to preserve data in the cells. To make matters worse, process variation magnifies the leakage problem. Since there is an exponential relation between leakage current and threshold voltage, increasing variations in threshold voltage can cause a 5X variation in leakage power across chips [21].

Process variation will negatively impact the speed, stability, and leakage of traditional SRAM designs. Solutions like simply sizing up devices have unacceptable area and power penalties. A desire to continue the scaling trend for on-chip memories in nanoscale technologies drives us to seek out new memory circuit and architecture solutions.

2.2 Introduction to 3T1D DRAM

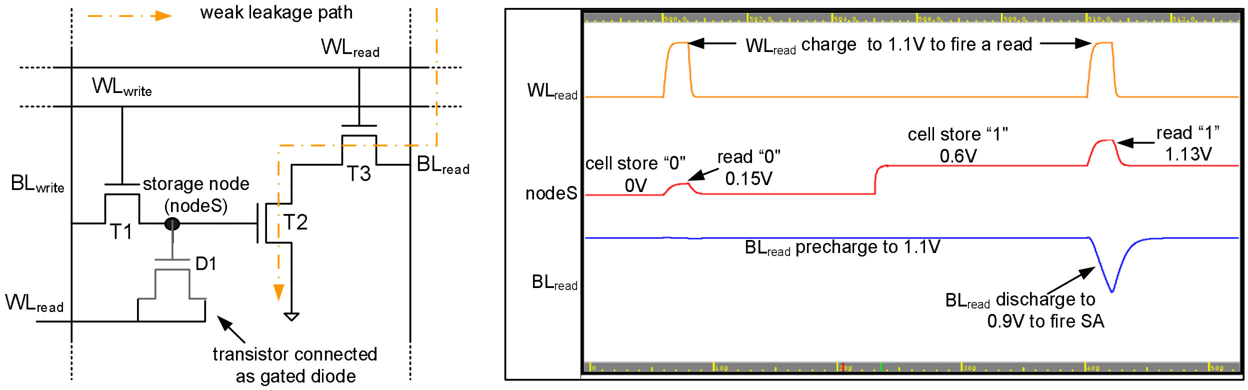
Recent circuit innovations in memory design provide an interesting alternative to 6T cells. Luk et al. proposed a novel 3T1D DRAM cell, which offers speed comparable to a 6T SRAM cell for a limited period of time after writing the data [18, 19]. Published results from chips fabricated in 130nm and 90nm technologies verify high-speed operation for dynamic 3T1D memories. In contrast, widely used 1T DRAM cells have slower access times and suffer from destructive reads. Hence, 3T1D cells are well-suited for latency-critical structures while 1T cells may be more appropriate for slower L2 caches. The proposed memory architecture in this



a. Leakage paths in a 6T SRAM cell

b. Read/write operation of a 6T SRAM cell

Figure 2. Traditional 6T-based SRAM cell design.



a. 3T1D DRAM cell and leakage path

b. Timing of read operation in 3T1D cell

Figure 3. 3T1D DRAM cell design and operation.

paper can scale more effectively with technology under increasing process variations by employing intelligent data retention schemes for these dynamic memories.

Figure 3(a) presents a schematic of the 3T1D (3-transistor, 1-diode) DRAM cell. Due to the threshold voltage of T1, there is a degraded level on the storage node when storing a "1". Hence, it relies on a "gated diode" (D1) to improve array access speed. This diode can be thought of as being a voltage-controlled capacitor with larger capacitance when storing a "1" and a smaller capacitance when storing a "0." Each time the cell is read, the bottom side of this capacitor is also raised to VDD. If the cell stores a "1" and it is read, the charge stored on the big capacitor of D1 boosts up the turn-on voltage of T2, rapidly discharging the bitline. As a result, the access speed can match the speed of 6T SRAM cells. Conversely, when a "0" is stored, the capacitance of D1 is smaller and there is almost no voltage boosting, which keeps T2 off during the read. Hspice simulation results, shown in Figure 3(b), illustrate the operation of the 3T1D cell. The gate voltage of T2 is boosted by about 1.5-2.5 times (1.13V) the originally stored voltage (0.6V) if a "1" is stored when being read.

Although the speed of a 3T1D cell can be fast, this high-speed access is only valid for a limited time period after each write to the cell. This is because the charge on D1 leaks away over time. Figure 4 shows the relationship between cell access time versus the elapsed time after a write operation. With this stored charge leaking away, the access time increases until finally it exceeds the array access time of the 6T SRAM cell. Traditionally, the word "retention time" is defined as the time a DRAM cell can no longer hold the stored value. But in this paper, we redefine the *retention time* of a 3T1D memory as the time period during which the access speed can match that of a 6T SRAM cell (from Figure 4, about 5.8μs for nominal cells). Within this retention time, the memory can be accessed at the nominal chip frequency. After this retention time passes, the memory cell has to be refreshed (re-written), otherwise that cell is considered invalid.

Process variation can also impact the 3T1D memory. For example, if the driving capability of the access transistors is reduced due to variations, access time of the cell increases. This effect can otherwise be viewed as decreasing retention time, which is shown in Figure 4. Weaker-than-designed access transistors have the effect of shifting the access time curve to the left and cell retention time reduces (down to 4μs). On the other hand, with stronger devices, retention time can increase. It is important to note that process variations do not necessarily impact operating frequency, which is the case for 6T SRAMs. In a 3T1D DRAM, process variability causes variations in retention time to achieve the same nominal access speed. Moreover, variations in the tags, in the speed of the decoder, sense amplifier, and other peripheral circuitry in the memory can be absorbed into this retention time variation. Thus, the impact of process variations on a 3T1D memory can all be lumped into a single variable—the retention time. As detailed in Section 4, we propose architectural techniques to tolerate very large retention time variations and, thus, effectively reduce the impact of device process variations in 3T1D-based memories. Furthermore, 3T1D cells do

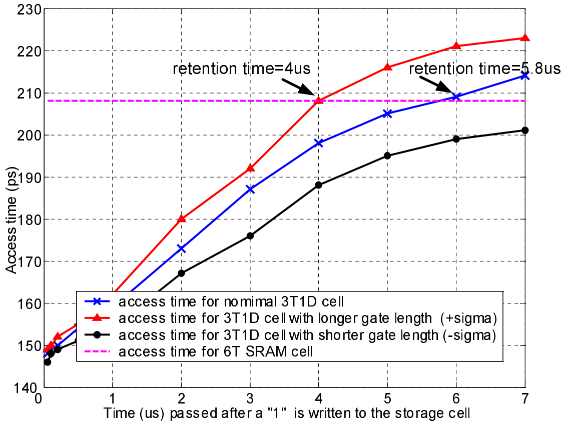


Figure 4. Relationship between access time and retention time in 3T1D cell.

not suffer the same type of cell stability issues previously seen for 6T SRAM cells, because there is no inherent fighting. Except for the finite data retention time, a 3T1D DRAM cell is stable.

The 3T1D-based memory array also offers advantages in terms of reduced leakage power since a 3T1D DRAM cell does not suffer the multitude of strong leakage paths previously seen in 6T SRAM cells. If there is a “0” stored in the cell, there is only one weak leakage path given two stacked off transistors, shown in Figure 3a. If there is a “1” stored, since most of the time the stored value is a degraded value (only boosted high for a short period during read), there is only one slightly strong leakage path. Furthermore, if the stored charge has leaked away, the slightly strong leaking path becomes weak. The smaller number of leakage paths leads to lower nominal leakage and less leakage variability.

3. Experimental Methodology

3.1 Circuit and Monte-Carlo Simulation

All the delay and power data presented in this paper is derived from Hspice circuit simulations. The minimum-size 6T SRAM cell is based on a design available from a commercial 130nm technology design library. In order to simultaneously accommodate 2 reads and 1 write, two additional access transistors with corresponding bitlines and wordlines are needed. To constrain cell size, we assume two single-ended reads and one differential write. While this results in 8 transistors per cell, the rest of the paper continues to refer to these SRAM cells as 6T SRAM cells (1X 6T) to avoid confusion with other 8T cell topologies. We designed the 3T1D DRAM cell (schematic and layout) with the same area as the 1X 6T cell. While the 3T1D cell can be smaller than a 1X 6T cell [18], the larger size (equal to 6T) improves data retention times. For comparison purposes, we also designed a larger 6T SRAM cell (2X 6T), where width and length of each transistor was sized to be 2 times larger than the devices in the 1X 6T cell. Based on these original designs, we then scaled the designs to 65nm, 45nm, and 32nm technology nodes. All Hspice simulations for the three technology nodes are based on Predictive Technology Models (PTM) [32]. All wires were scaled with respect to technology and cell area. We assume copper wires and use distributed-π models for wire delay simulation. Table 1 presents the detailed circuit parameters of our design and the simulation temperature is set to 80^o C. All of the simulation plots in this paper are based on the 32nm technology, although we also present results of other technology nodes in the tables.

We rely on Monte-Carlo simulations for variation analysis, similar to approaches found in [1, 17, 21]. This method considers both die-to-die and within-die variations, and also handles correlations related to layout geometries using a 3-level quad tree method. Correlated variations affect access time, but in nanoscale technologies, increasing dopant fluctuations exacerbate random variations, causing access time differences between cells and compromising stability [2]. Recent experimental results verify that this method is very accurate; it has an error of 5% [7], which is sufficient for our architectural study. We modeled both random dopant and correlated gate length variations. Based on Friedberg’s chip measurements [9], we assume the gate length of all the transistors in each small sub-array are strongly correlated.

We consider two situations for variation in this paper. The *typical* variation assumes $\sigma L/L_{nominal} = 5\%$ for within-die gate-length variations and $\sigma V_{th}/V_{th_{nominal}} = 10\%$ for threshold voltage variations. The *severe* variation assumes $\sigma L/L_{nominal} = 7\%$ for within-die gate-length variations and $\sigma V_{th}/V_{th_{nominal}} = 15\%$ for threshold voltage variations. For both situations, we assume $\sigma L/L_{nominal} = 5\%$ for die-to-die gate length variation. These assumptions are comparable to the data forecast in [4, 5].

3.2 Architecture Simulation

We assume a baseline machine with parameters listed in Table 2, which is comparable to the Alpha 21264 and POWER4. The data cache is a 64KB, 512-bit block size, 4-way set associative, write-back memory, with 2 read ports and 1 write port. This cache is divided into 8 sub-arrays of 256x256b with a cache access latency of three cycles where one cycle is reserved to access the array. Every pair of arrays share 64 sense amplifiers and combine to form the 512-bit blocks. For our IPC simulations, we utilize the *sim-alpha* simulator [8].

We run an exhaustive set of architecture simulations to investigate the system-level impact of process variations. For example, each data point of Monte-Carlo simulation requires a corresponding architecture simulation in order to quantify the system-level impact of the variations. To manage this large number of simulations, we use 8 of the 26 SPEC2000 benchmarks and rely on SimPoint for sampling [27]. Phansalkar et al. show that these 8 benchmarks (*crafty, applu, fma3d, gcc, gzip, mcf, mesa, twolf*) can adequately represent the entire SPEC2000 benchmark suite [24]. For each benchmark, 100 million instructions are simulated after fast forwarding to specific checkpoints. When we report single number results (performance or power) in this paper, they represent the harmonic mean of the 8 simulated benchmarks.

4. Process Variation Tolerant Cache Architectures

Based on our earlier discussion of 3T1D memory cells, this section investigates the architectural support needed to allow 3T1D cells to replace traditional 6T cells. 3T1D cells provide many benefits compared to traditional 6T memory cells, but present challenges that must be overcome with system-level support. The major issue associated with 3T1D memories is the limited data retention time and variations of this retention time. Given the relationship between access latency and retention time, the cell can only hold the data for a short time period to ensure the fast-access requirement of on-chip memories. This paper discusses several approaches to accommodate this limited retention time with periodic data refreshing, retention-time driven replacement policies, allowing data to expire (no refresh), and combinations thereof.

Technology node	Min. size cell area for cache	Wire width	Wire thickness	Oxide thickness	Chip frequency
$65nm$	$0.90um^2$	$0.10um$	$0.20um$	$1.2nm$	$3.0GHz$
$45nm$	$0.45um^2$	$0.07um$	$0.14um$	$1.1nm$	$3.5GHz$
$32nm$	$0.23um^2$	$0.05um$	$0.10um$	$1.0nm$	$4.3GHz$

Table 1. Parameters for circuit simulation.

Configuration Parameter	Value	Configuration Parameter	Value
Issue Width	4 instructions	Issue Queues	20-entry INT, 15-entry FP
Load Queue	32-entries	Store Queue	32-entries
Reorder Buffer	80-entry	I-Cache, D-cache	64KB, 4-way Set Associative
Instruction TLB	128-entry Fully-Associative	Data TLB	128-entry Fully-Associative
Integer Functional Units	4 FUs	Floating Point Functional Units	2 FUs
L2 Cache	2MB 4-way	Branch Predictor	21264 Tournament Predictor

Table 2. Baseline processor configuration.

To present a clear view of our scheme, we first study the 3T1D cache without considering any variations. Cell-to-cell variations in retention times required by each 3T1D memory cell exacerbate the retention time problem. Hence, we investigate several variation-tolerant strategies.

4.1 3T1D cache without considering process variation

The simplest way to provide support for data retention in 3T1D caches is to use a global refresh scheme. Refresh operations require a read and a subsequent write back to the memory. A rudimentary refresh mechanism would add an extra read/write port specifically dedicated for refresh. However, this approach suffers considerable area and power overhead. Instead, as shown in Figure 5(a), we opt for less costly refresh mechanisms that leverage existing ports in the memory used for normal access. Whenever a refresh is needed, one read and one write port are blocked from normal cache accesses and are used to refresh the data. The refresh mechanism pipelines these read and write accesses on consecutive cycles. However, this approach introduces a performance penalty, because the refresh operation competes with normal instructions. We encapsulate the refresh into each sub-array of the cache. For a 512-bit cache line, we refresh 64 bits per cycle, limited by the number of sense amplifiers in the cache. In parallel, the tag array can be refreshed as well. It takes 8 clock cycles to refresh one cache line, thus, requiring 2K cycles to complete the refresh (for a 256-line sub-array). In our 32nm 4.3GHz machine, this requires 476.3ns. A global counter is used to generate a global refresh signal according to the retention time of the cache. During refresh, one read and one write port are blocked and this block signal is passed back to the scheduler to delay normal load/store operations.

Our circuit simulation shows the cache retention time is about 6000ns under the 32nm technology node. This means the refresh only takes about 8% (476.3ns/6000ns) of cache bandwidth. Because cache traffic is usually no more than 30% on average, the under-utilization of the architecture can be leveraged to hide refresh operations and result in negligible performance loss. Our detailed architecture simulations report less than 1% performance loss for such a configuration. Also, there is only one global counter needed for refresh and hence the hardware overhead is negligible.

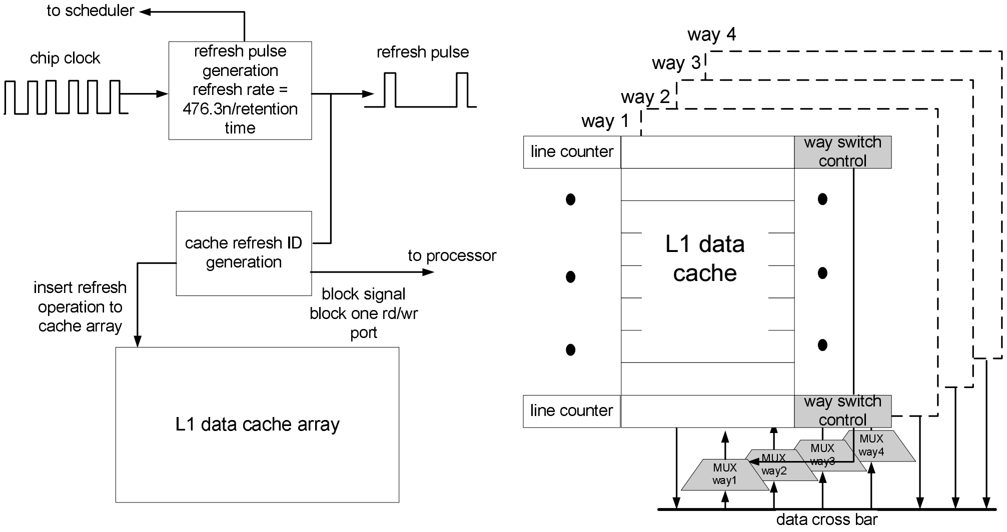
4.2 Dealing with typical variation, global refresh scheme

We first consider the impact of process variations on traditional 6T SRAM caches. Figure 6(a) shows the performance of the 6T SRAM cache with typical process variations. With 1X 6T cells, the frequency and hence performance for most chips lose about 10% to 20% compared to the ideal design. Even when using the 2X 6T cells, 20% of chips still suffer about 3% of performance loss (the .975 bar in the plot).

In contrast, process variations introduce retention time variations in the 3T1D caches. For each fabricated cache, the memory cell with the shortest retention time determines the retention time of the entire structure. The top plot in Figure 6(b) presents a histogram of retention time distribution for the 3T1D cache given typical process variations. Although all of the chips are designed with the same parameters, they exhibit a wide retention time spread due to process variations. However, the middle plot of Figure 6(b) shows that the processor performance only varies by about 2% with retention time varying from 714ns to 3094ns. Significant performance loss is observed only when retention time falls below 500ns. In this figure, 97% of the 3T1D caches only lose less than 2% of performance compared with an ideal 6T design. Even for the worst-performing benchmark (fma3d), the performance penalty is less than 4%. The resulting insight is that out-of-order processors can tolerant large retention time variations. Since physical device variations cause retention time variation in the caches, the architecture of the processor can help to mitigate the impact of process variations.

Therefore, compared to a SRAM design, a 3T1D cache achieves much better performance for comparable yields. However, this increased performance comes at the expense of additional dynamic energy consumption due to refresh. The dynamic power overhead is from 1.3-2.25X of the ideal 6T design, shown in the bottom plot of Figure 6(b).

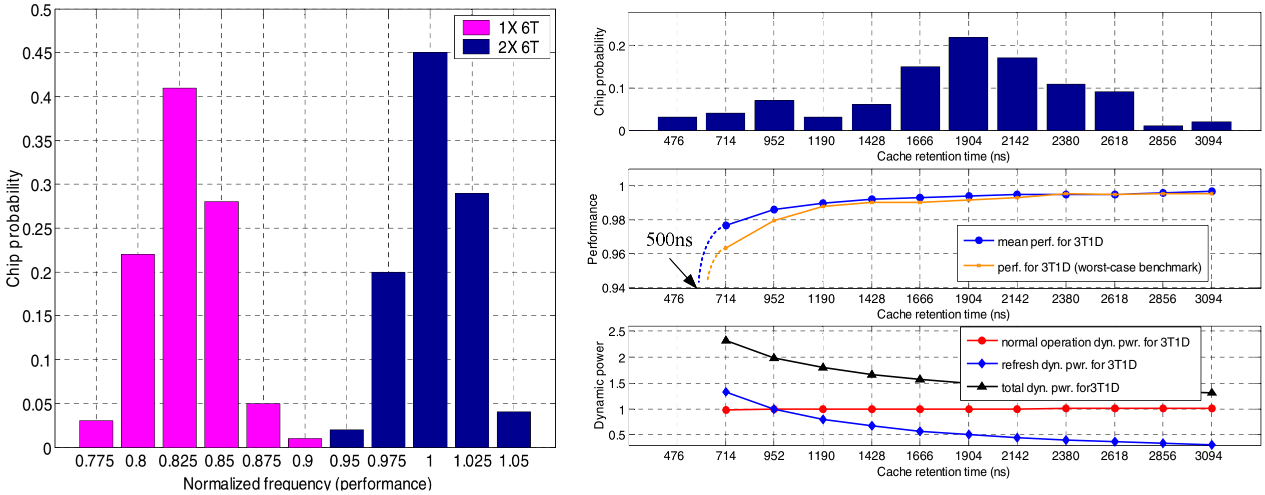
While dynamic power may be larger, leakage power tends to dominate cache structures at 32nm. The advantages of 3T1D for



a. global refresh scheme for cache

b. line-level refresh schemes for cache, shaded blocks only used in RSP-FIFO and RSP-LRU schemes

Figure 5. Proposed cache architectures for global and line-level schemes.



a. 6T cache frequency/performance distribution (1X 6T, 2X 6T)

b. 3T1D cache retention time distribution, performance and dynamic power

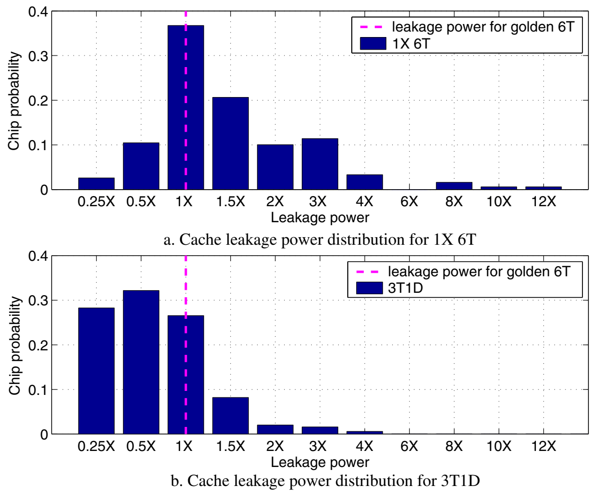
Figure 6. Distribution, performance and power for 6T and 3T1D caches

leakage become more prominent. As a result, the lower leakage afforded by 3T1D cells leads to an overall reduction in cache power. Figure 7 compares the distribution of leakage power in the presence of process variations. For the 1X 6T cache, more than 50% of the chips exhibit leakage power greater than 1.5X of the original design and some chips show more than 10X leakage. For the 3T1D cache, only about 11% of the chips have leakage greater than that of the golden 6T design and it never exceeds 4X. Realistic dynamic power numbers derived from access patterns given by the architectural simulator for our cache structure show that leakage power is significantly higher than dynamic power at the 32nm technology node. Given this disparity, the 3T1D's leakage power reduction far outweighs the small increase in dynamic power. It is important to note that predictive technology models do not include advanced leakage-mitigation technologies such as metal gates with high-K dielectrics introduced at the 45nm node. For comparison, our leakage power results are comparable to the POWER6 in the 65nm node where leakage is shown to be as high as 42% [10].

Table 3 provides a detailed summary of simulation data for the cache designs across different technology nodes. Comparing the performance of 1X 6T design with the ideal SRAM design, there is approximately one generation of performance loss. In contrast, the performance of 3T1D scales well and there is still room for future scaling. Typical 3T1D chips can also save about 64% of cache power consumption compared with the ideal 6T SRAM design.

	ideal 6T, no variation					1X sized 6T, typical variation, median chip					3T1D, typical variation, median chip				
Tech. node	Access time	Perf. (BIPS)	Mean Dyn. Pwr	Full Dyn. Pwr	Leakage Power	Access time	Perf. (BIPS)	Mean Dyn. Pwr	Full Dyn. Pwr	Leakage Power	Retention time	Perf. (BIPS)	Mean Dyn. Pwr	Full Dyn. Pwr	Leakage Power
$65nm$	$285ps$	2.91	$4.30mw$	$31.97mw$	$15.8mw$	$370ps$	2.33	$3.44mw$	$25.29mw$	$15.8mw$	$4000ns$	2.89	$5.98mw$	$29.93mw$	$3.36mw$
$45nm$	$251ps$	3.39	$3.41mw$	$25.96mw$	$36.0mw$	$315ps$	2.82	$2.83mw$	$21.55mw$	$36.0mw$	$2900ns$	3.37	$4.85mw$	$24.65mw$	$5.68mw$
$32nm$	$208ps$	4.17	$2.78mw$	$20.75mw$	$78.2mw$	$251ps$	3.50	$2.39mw$	$19.21mw$	$78.2mw$	$1900ns$	4.14	$4.08mw$	$20.30mw$	$24.4mw$

Table 3. Detailed simulation data of cache for three technology nodes



a. Cache leakage power distribution for 1X 6T

b. Cache leakage power distribution for 3T1D

Figure 7. Cache leakage power distribution with process variations

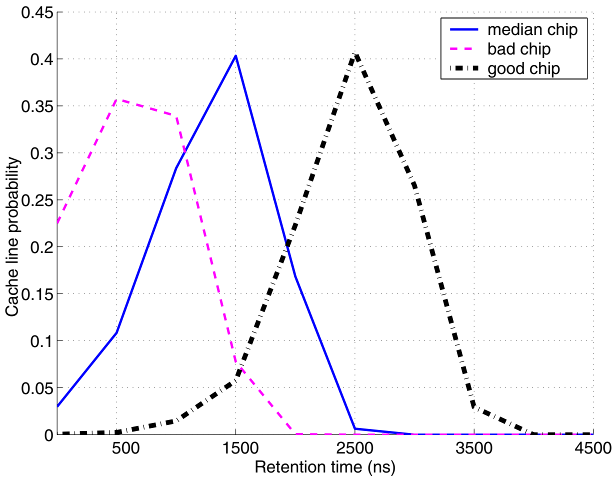


Figure 8. Retention time distribution of cache lines for best, worst and median chips under severe variations.

4.3 Dealing with severe variation, line-level schemes

One major disadvantage of the 6T SRAM cache is that the speed of the slowest cells determines the operating frequency of the entire structure. Due to clock generation and synchronization limitations, it is very challenging to access each cache line using different clock frequencies. The proposed global refresh scheme for the 3T1D cache suffers a similar disadvantage, because the worst memory cell determines the retention time of the entire structure. Although the global refresh scheme works well for typical process variations, it loses its effectiveness given severe process variations. To illustrate this problem, we select three specific chips for further analysis: one “good” chip with process corners that result in longest retention time, one median chip, and one “bad” chip with process corners that result in shortest retention times. We plot histograms of the retention time of cache lines for the three chips, shown in Figure 8. This plot illustrates the wide retention time distribution among different cache lines even within a single chip. Up to 23% of the cache lines in the bad chip and 3% of the cache lines in the median chip are considered to be “dead lines,” because their retention time is zero. These dead cache lines cannot function properly under severe variations. Retention time of the whole cache in the global scheme is limited by the worst cache lines, so for both the bad and the median chips, the retention time has to be set to zero and those chips must be discarded. Our simulations show that about 80% of the chips must be discarded, because they contain at least one line that cannot function due to very short retention times. This inefficiency arises because the global scheme is too coarse grained and cannot distinguish differing retention time requirements of each cache line. Due to significant within-die variations, retention times can differ greatly across the cache lines. To address this spread, we need to find more fine-grained schemes that can avoid this “one-kill-all” situation. Also, explicit global refresh introduces significant dynamic power overhead to the system, which ought to be reduced. Since the operating frequency can remain constant for 3T1D memories and only retention time varies, it is much easier to apply line-level schemes compared with traditional SRAMs.

4.3.1 Line-level Refresh Policies

In this section, we evaluate line-level refresh schemes (word-level refresh is also possible, but is not studied due to the excessive hardware overheads). Line-level refresh requires each line to be tagged with a retention time (which is defined by the lowest retention time of the cells in that line, so that no data is lost during the retention time). In terms of hardware budget, line-level refresh requires an extra counter per line to individually keep track of retention times. The line counter is reset once a new data block is loaded to the cache line and starts counting. Once it counts to the line retention time, that line is no longer useful and needs to be refreshed or evicted depending on different schemes. All line counters are synchronized to a global clock, which is a fraction $(1/N)$ of the chip frequency. This means the minimal time step of the line counter is N cycles. N can be set according to difference variation conditions. For example, larger retention time requires larger N so that for the counter with the same number of bits, it can count more. We estimate that the 3-bit line counters add about 10% hardware overhead to the design and the power overhead is shown later. After fabrication, the retention time of each individual cache line must be determined and stored in the line counters. To test the cache, a built-in self test structure can load a pattern of "1s" into the cache and keep reading out the contents of each line until the line fails to give correct value. The amount of time required to fail reading the "1s" pattern is recorded as the line retention time. Although dynamic testing is possible, we assume worst-case temperatures to set retention times in this paper.

Refreshing the cache lines will increase their lifetime, which can be beneficial to system performance. On the other hand, frequent refreshes block normal cache accesses and reduces the effective bandwidth, which can degrade system performance. To further study this problem, we consider a wide spectrum of possible refresh policies. These range from from refreshing each individual line to completely avoiding refresh and relying on the inclusion properties of the cache to recover data from the L2. We also consider partial refresh schemes that are a hybrid of these two schemes. We have identified three schemes in the spectrum:

No-refresh: We evaluate a technique that does not refresh any cache lines. Thus, whenever the retention time of that cache line reaches a low threshold, the line is evicted. While write-through caches do not require any action, write-back caches require dirty data to be written to the L2 cache. In the pathological scenario where many dirty lines expire simultaneously, write-back traffic can cause the write buffer to stall. To ensure data integrity, dirty lines waiting for eviction are refreshed during this stall. The only hardware overheads are the line counters and control logic.

Partial-refresh: This technique tracks the retention time of each line. If the retention time is below a specific threshold, that line will be refreshed and it will stay alive until passing the threshold time. Any lines with retention time larger than the threshold will not be refreshed. This means that just a subset of the lines are refreshed depending on their distinct retention times. This scheme guarantees that all of the lines have a lifetime longer than the threshold before they are evicted. Whenever a line requires refresh, it asserts a signal for refresh. It is possible that multiple lines will assert for refresh at the same time. To handle these cases, we implement a token scheme which iterates through lines tagged for refresh in the cache. A line can obtain refresh only when it asserts the refresh signal and has the token. This scheme has the possibility of delaying the refresh of some lines that do not receive the token immediately. To ensure data integrity, we conservatively set the retention time counter to guarantee each line will receive the token before expiring. Partial refresh requires the one bit token and extra control logic to generate the refresh assertion signal, which only require 3-4 logic gates. The complexity of this scheme is slightly higher than the no-refresh scheme.

Full-refresh: All lines of the cache will always be refreshed before the retention time expires. The refresh scheme is similar to the partial-refresh scheme. The only difference is that all of the cache lines are refreshed and there is no threshold time.

4.3.2 Replacement Policies

The cache line replacement policy also provides an important approach to support the data retention requirements of the 3T1D cache. In a conventional cache, typically a LRU (Least Recently Used) replacement policy is implemented. This policy always selects the least used block, which may not be the best choice for 3T1D caches. Under process variations, lines within each set may have different retention times, and if the no-refresh scheme is implemented, then it is very likely that the lines with smaller retention times may be "free" most of the time (since their data are evicted more regularly than lines with longer retention times). As seen from the line usage of the benchmarks in Figure 1, we can see that on average, most of the accesses (90%) to a line are concentrated in its first 6K cycles of lifetime. Thus, it is likely not effective to assign the newly-loaded data to the lowest retention time way since the data may not be retained in the cache for sufficient time. Thus, we propose three replacement policies that are sensitive to the retention time requirements of 3T1D-caches.

Dead-Sensitive-Placement (DSP): We know that some cache lines may have zero retention time and are "dead" due to within-die variations. In this paper, we also treat a cache line as dead if its retention time is less than the minimal time step (N) of the line counter. This policy replaces lines in a manner that avoids using the dead lines. When replacing, the lines are handled similar to a conventional LRU, but no dead lines will be used. For example, in a 4-way cache with one dead line, the replacement policy will always make use of the other three ways. In the worst case, given a set with all ways dead, those accesses/replacements will miss the L1 cache and incur a corresponding L2 access. Not being aware of the dead lines can significantly degrade system performance, because those dead lines will be mistakenly treated as being useful and the processor will take for granted that there is valid data in the dead lines. This can lead to a large number of cache misses and increase the occurrences of replay and flush in the pipeline.

Retention-Sensitive-Placement-FIFO (RSP-FIFO): This policy logically organizes the lines in each set in a FIFO scheme. The order of the lines in the FIFO is in descending retention times. Thus, "new" blocks are assigned the longest retention time line, and the blocks in that set are moved to the next longest retention time line. The data held in the lowest retention time line is evicted. This scheme relies on the theory that most cache accesses happen within the initial period after the data is loaded. Moving newly loaded data into longer retention time locations can greatly reduce the number of cache misses resulting from expired retention times. This mechanism intrinsically incorporates a refresh since every new block inserted into the cache results in the existing blocks in that set moving to new lines (with smaller retention times). When accessing the cache, all ways are read; thus, the overhead includes that of writing back those lines into a different way. This operation requires multiplexors so that data can be switched between the different ways, as depicted in the shaded blocks in Figure 5(b). For example, in our 4-way cache design, we use a 4-to-1 MUX. Besides the multiplexors, we need to configure switch control registers for each cache set at test time, since the variations of each cache set can be different. Our circuit estimates anticipate an additional 7% hardware overhead for the this logic. It takes 8 cycles to move a 512-bit line with 64 sense amplifiers.

Retention-Sensitive-Placement-LRU (RSP-LRU): This technique is similar to the previous one, but in this case the most-recently-accessed block is kept in the longest retention time line. Thus, every read or write (hit or miss) may shuffle lines among the ways in order to keep them in the appropriate order (highly active blocks reside in the higher retention time lines). This scheme relies on the theory that data accessed in the cache will likely be accessed again in the near future (temporal locality). This scheme keeps the most recently access data in the location with longest retention time, improving the possibility of a cache hit. This scheme is complex because the line shuffling occurs more frequently compared to RSP-FIFO, which only shuffles lines on cache evictions.

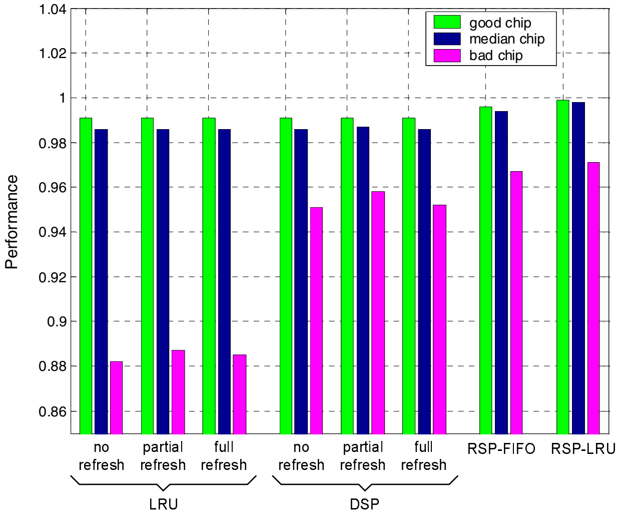


Figure 9. Normalized performance of retention schemes for good, median and bad chips.

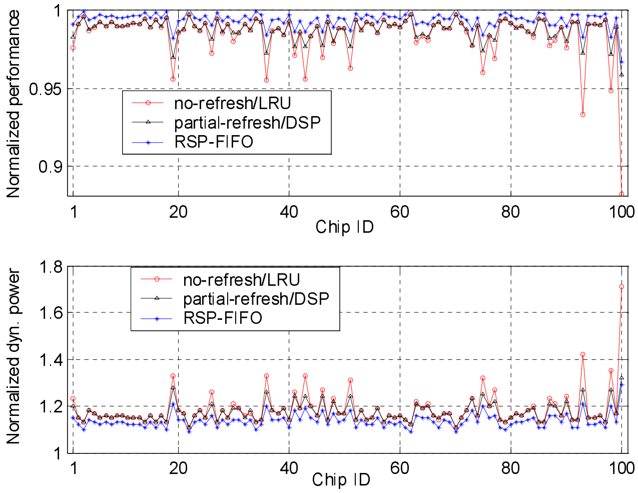


Figure 10. Normalized performance and power of 100 chips for three line-level schemes (sorted by descending performance of no-refresh/LRU).

These replacement policies are orthogonal to the previously proposed refresh policies. We now discuss hybrid combinations of the refresh policies and replacement policies.

4.3.3 Combining Refresh and Replacement Policies

Given the refresh policies and replacement polices, in this section we analyze interesting combinations of the aforementioned policies. The cross-product of the three refreshing schemes (no-refresh, partial-refresh, full-refresh) and the four replacement policies (LRU, DSP, RSP-LRU, RSP-FIFO) would provide a total of 12 techniques to evaluate. However, we can reduce that set to 8 if we consider that the placement policies RSP-LRU and RSP-FIFO do not need to be augmented with a refresh policy since they already implement an intrinsic refresh (i.e. when moving lines). Thus, we will evaluate the previously cited RSP-LRU and RSP-FIFO, plus the remaining six combinations: no-refresh/LRU, partial-refresh/LRU, full-refresh/LRU, no-refresh/DSP, partial-refresh/DSP and full-refresh/DSP.

We generated 100 sample chips under severe process variations. To evaluate the efficiency of all schemes, we pick the same three chips discussed in Figure 8 for analysis. We have analyzed the performance of these chips for the eight schemes described above and plot the performance relative to an ideal 6T cache design in Figure 9.

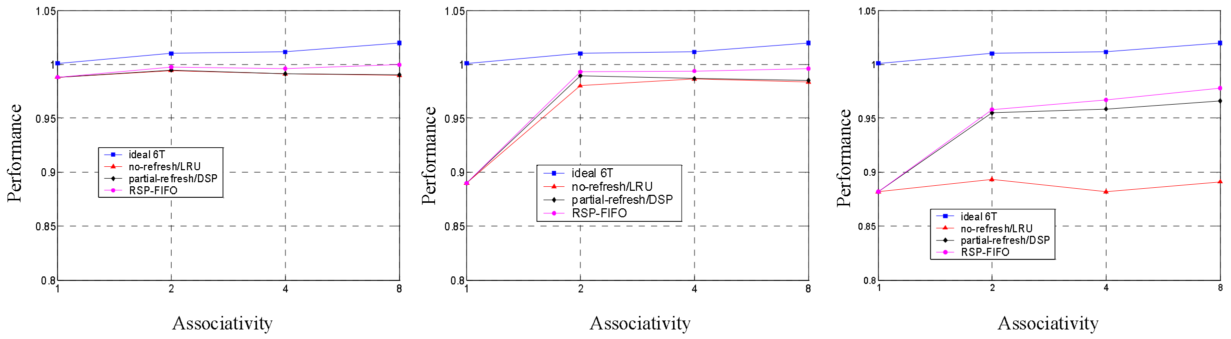
The performance difference for the different line-level retention schemes is most relevant for the worst chip. In this case, the techniques that only implement the conventional LRU replacement policy suffer significant performance losses due to a large number of dead line references. This behavior is also observed for the good and median chip, but to a much smaller extent. Recognizing the presence of these dead lines can significantly reduce cache misses due to the unwanted accesses to invalid lines and avoid the subsequent pipeline replay. At the same time, we can see that the partial-refresh mechanisms increase performance when available by 1-2% (e.g. no-refresh/LRU vs. partial-refresh/LRU, and no-refresh/DSP vs. partial-refresh/DSP). If the number of lines refreshed is extended to the entire cache (full-refresh), then the effect of blocking loads and stores for refreshing the cache introduces a performance penalty (1%). Thus, performance decreases when moving from the partial-refresh schemes to the corresponding full-refresh schemes. The results provide an important insight: cache refresh may not always improve performance. Only refreshing lines with short retention time while keeping other lines intact often outperforms full-refresh. Finally, we see that the combination of a placement policy with an intrinsic refresh scheme (RSP-LRU and RSP-FIFO) produce the most performance benefits. The advantage comes from the more intensive use of cache lines with longer retention time, which can help to reduce the average miss rate of the 3T1D cache.

To simplify the remainder of the analysis in this paper, we choose one technique for each placement scheme: no-refresh/LRU, partial-refresh/DSP, and RSP-FIFO. These techniques represent a broad spectrum of combinations. The first (no-Refresh/LRU) represents the simplest line-level scheme and requires the least amount of modification to a conventional cache design. The second (partial-refresh/DSP) scheme has the benefit of avoiding dead lines and refreshing a subset of the lines. These results use a 6K-cycle threshold for the partial-refresh scheme. The third scheme, RSP-FIFO, represents the family of more complex schemes but, at the same time, offers the best performance.

4.3.4 Detailed Evaluation of the Schemes

Each of the three schemes rely on different characteristics of the cache architecture and program behavior. For comparison, we perform detailed performance and power simulations with our cycle-accurate architecture simulator. For this analysis, we use the entire set of 100 randomly generated chips suffering severe variations.

Figure 10 (upper) plots the performance of all chips using the three schemes. All of the chips can still function even under severe variations with a small performance penalty. In contrast, 80% of the chips must be discarded when using the global scheme under the same amount of variations, and the 6T cache would suffer a 40% frequency loss. Furthermore, almost every cache line in a conventional 6T cache would contain unstable cells given this amount of process variations. This amount of instability would be difficult to overcome using redundancy and ECC mechanisms. In comparison, the line-level retention policies are able to overcome the large amount of variations in the memory.



a. good chip b. median chip c. bad chip

Figure 11. Performance of good, bad and median chips using three line-level scheme with different associativities.

Compared with the other two schemes, no-refresh/LRU scheme incurs the largest performance penalty. This is due to the fact that the no-refresh/LRU scheme tracks the retention time for each cache line, but does not include any special refresh or replacement policy. On the other hand, no-refresh/LRU scheme has the least hardware overhead and control mechanism among the three schemes.

Both RSP-FIFO and partial-refresh/DSP schemes have superior performance over the other schemes with a performance penalty of less than 3% for all 100 chips. In fact, most of the chips incur less than 1% performance loss. Although the two schemes rely on different insights, both schemes track the retention time variation in the different cache ways. Partial-refresh/DSP avoids using the dead ways in a cache set, while RSP-FIFO will always make most use of cache ways with longer retention times. Compared with partial-refresh/DSP, RSP-FIFO is a finer-grained scheme because it distinguishes the different retention times of each cache way (more information stored) while partial-refresh/DSP only knows whether a cache way is dead or not (less information stored).

To illustrate the importance of cache parameters on the performance of the three schemes, we picked the same good, median, and bad chips in the previous section and plot the performance for different associativities in Figure 11. The performance difference between the three schemes is small for good and median chips. But for bad chips, suffering significant variations, RSP-FIFO and partial-refresh/DSP significantly outperform no-refresh/LRU for associative caches. We see that 2- and 4-way set associative caches are able to provide enough flexibility to accommodate the dead lines and for retention-sensitive replacement schemes to function properly. For direct-mapped caches, the replacement polices have no effect and, thus, the only performance benefits arise from the refresh policies.

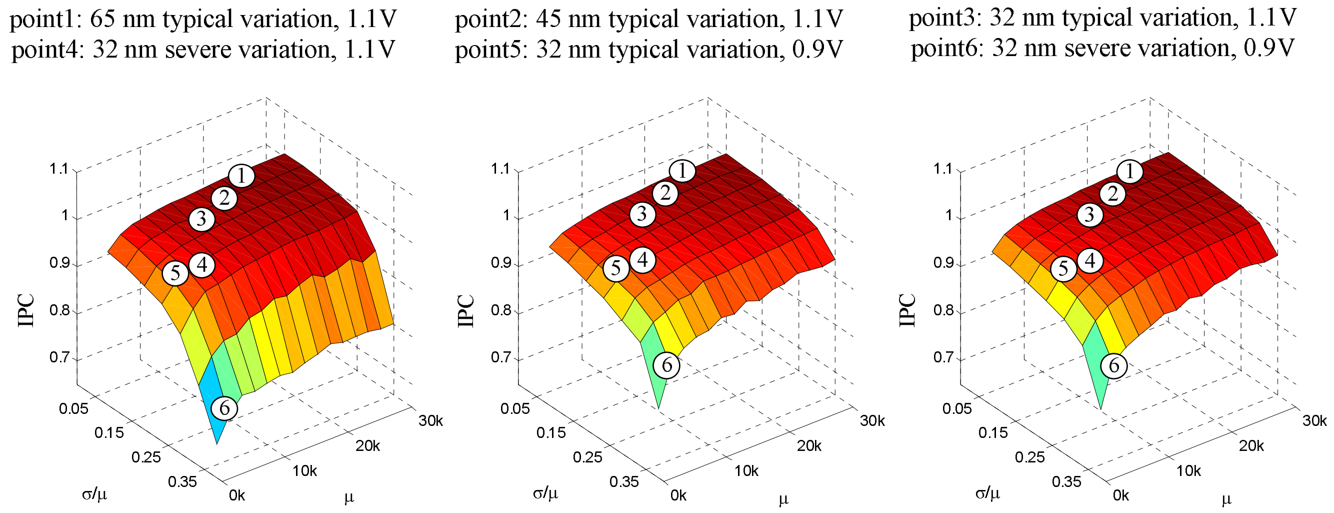
The line-level retention schemes all include mechanisms to reduce the overhead that the global refresh scheme incurs by continuous refresh. Figure 10 (bottom) plots the dynamic power consumption of the three line-level schemes. These power estimates include all power overheads incurred for the schemes and is normalized to the dynamic power consumption of an ideal 6T cache. For example, the no-refresh/LRU scheme consumes additional power for line counters and may incur additional L2 cache accesses when a line's retention period expires and the data is required again. The power overhead for the no-refresh scheme is less than 20%, while for some bad chips (e.g., chip #100), the overhead can be as high as 60%. Many of the cache lines in the bad chips have very short retention times and incur frequent L1 misses, which consequently increases the power overhead of extra L2 accesses. In contrast, we find that the chips using the RSP-FIFO and partial-refresh/DSP schemes have less than 10% power overhead. Although both of these schemes incur overheads from line counters and refresh power, they make more efficient use of cache ways with longer retention times, greatly reducing L1 misses. The RSP-FIFO policy also incurs a power overhead to switch or move data blocks between the different ways when necessary. Compared to the global scheme, which has 30%-125% dynamic power overhead, all the schemes shown here are much more efficient in terms of dynamic power, and enjoy leakage power benefits similar to those discussed in Section 4.2

5. Sensitivity Study

In the previous section, we showed the effectiveness of the cache architecture under typical and severe process variations at the 32nm technology node. In order to gain a thorough understanding of the impact of retention time variation on system performance, we sweep different μ and σ/μ of retention time, where μ is the mean retention time of all the cache lines in a chip and σ is the standard derivation of the retention time. To simplify the problem, in this section, we will not consider die-to-die variations but only focus on within-die variation. We only consider retention time variations among cache lines in one single chip.

Figure 12 presents surface plots of system performance for different μ and σ/μ combinations. We assume the mean retention time μ can change from 2K to 30K clock cycles, and σ/μ can vary from 5% to 35%. Depending on the process technology, supply voltage, and severity of process variations, a real design can be located at different points on the surface plot.

We can draw several conclusions from the Figure 12. First, we see that σ/μ has a larger impact on system perfromance than μ. In other words, variance of retention time has larger impact than the mean retention time. The large performance penalty is mostly due to the number of dead lines in a particular chip. The number of dead lines increases quickly with larger variance, making the cache less efficient. There is a sudden performance drop with σ/μ larger than 25%, which means the proposed cache can tolerate large retention time variations among different cache lines. Second, for the same σ/μ, larger μ helps to improve the performance. Larger μ means the active lines can have a larger retetion time so they can meet the system requirement with less or no refresh. Third, dead line sensitive and retention sensitive schemes behave much better than a no-refresh scheme for almost all of the points on the plot. The difference between dead line sensitive and retention sensitive is minor. However, since dead line sensitive schemes have smaller hardware overhead, we consider them most hardware efficient.



a. no-refresh/LRU

b. partial-refresh/DSP

c. RSP-FIFO

Figure 12. μ-σ/μ-performance plot for three schemes

In this figure, we also marked some real design points so that readers can get a feeling for how this plot corresponds to real designs. For example, point 4 corresponds to a chip implemented in a 32nm technology suffering severe process variations with a 1.1V supply. Point 2 corresponds to a chip implemented in a 45nm suffering typical variation with a 1.1V supply. Process, supply voltage, and the amount of process variations will change the σ and μ of retention time and, thus, affect system performance. We can see the trend of how performance varies with respect to the process technology, voltage, and amount of variation. For example, point 1, point 2, and point 3 show performance somewhat degrades with technology scaling since retention times decrease. Point 3 and point 5 reveal that scaling voltage to lower levels also impacts retention times and degrades performance. Clearly, as the severity of process variations gets worse, performance also suffers. The proposed cache architecture can achieve satisfactory performance for most of the points, but under worst-case assumptions (point 6) there can be considerable performance loss. Hence, continued innovations at all layers of design (process technology, circuit design, architectural techniques) are needed to push future chips to the upper corner. For example, continued development and progress in process technologies to address leakage power concerns, such as metal gate devices with high-K dielectrics, can also significantly improve 3T1D mean retention time and variance. It is important to point out that severe conditions, such as point 6, would prevent conventional SRAM caches from achieving comparable performance.

6. Related Work

In recent years, process variation has been identified as one of the key threats to continued Moore's Law scaling, with projections that a technology generation of performance can be lost due to process variations [5]. Also, there are serious concerns about the continued scalability of SRAM-based memories [3]. Several groups have proposed solutions to patch stability issues due to process variations in memory designs that use 6T SRAM cells [14, 25].

Recently, researchers have begun to explore the system-level impact of variations on power, performance, and reliability. Initial work in this area has focused on the modeling of process variations [26]. Researchers have shown that the selection of pipeline depth [4, 15] and other microarchitectural parameters, at design time, can significantly impact the susceptibility of an architecture to process variations. Variable-latency techniques have been proposed for caches, register files, and pipelined logic structures [22, 17, 29]. Globally-asynchronous, locally synchronous (GALS) design techniques may offer ways to mitigate the impact of correlated within-die variations [20]. Agarwal et al. propose to resize caches in response to variation-induced failures after fabrication [2].

The integration of 1T DRAM memories within microprocessors has been proposed as a means to increase the effective bit density of on-chip cache memories [28, 16, 23]. In these approaches, bit density is the primary goal and the use of 1T memories and traditional refresh policies is sufficient, although Lee et al. [16] propose a selective invalidation scheme for a 1T-DRAM based instruction memory allowing portions of the memory to be invalidated if not read or written during a time period. The concept of utilizing transient data has been applied to low-power design. The RAPID project uses a software approach to reduce the refresh power of commodity off-the-shelf DRAMs by allocating memory to pages of DRAM which require less frequent refresh operations [30]. The decay cache provides a fine-grained leakage control scheme by enabling Vdd-gating on a per-line basis after specified time intervals [13]. Juang et al. recognized that quasi-static 4T cells could be used to avoid the need to implement Vdd-gating transistors and demonstrate power savings benefits in a decaying branch predictor [12]. None of the above works consider the impact of process variations and, hence, they do not consider the variety of line-level retention time mechanisms that we consider. Our work proposes to replace on-chip SRAM with 3T1D DRAM memories, with the specific target of combating process variations. The proposed memory architecture provides advantages in terms of cell stability, reduced power requirements, and the ability to tolerate performance variations by intelligently leveraging the computer architecture. This approach provides a comprehensive solution to many of the issues that will impact on-chip memory designs in nanoscale process technologies.

7. Conclusion

Microprocessors tolerant to process variations in future nanoscale technologies will be at the forefront of innovation for years to come. This paper proposes novel process variation tolerant on-chip memory architectures based on a 3T1D dynamic memory cell. The 3T1D DRAM cell is an attractive alternative to conventional 6T cells for next-generation on-chip memory designs since they offer better tolerance to process variations that impact performance, cell stability, and leakage power.

Two categories of schemes (refresh and placement) have been proposed for one the most important on-chip memory structures – the L1 data cache. By leveraging under-utilization of architectural resources, transient data characteristics, and program behavior, significant performance and power benefits have been demonstrated for 3T1D memories in comparison to 6T SRAM memories. Under typical variations, the proposed schemes perform within 2% of the performance assuming ideal 6T memories. Under severe variations, where 6T SRAM caches would suffer a 40% reduction in frequency, the schemes presented can perform with less than 4% performance loss and offer inherent extra benefits of reducing leakage power. These promising results suggest that 3T1D memories can replace existing on-chip 6T memories as a comprehensive solution to deal with process variations.

Acknowledgments

This work was supported by NSF grants CCF-0429782 and CCF-0702344; the Fulbright program; the Generalitat de Catalunya (2005SGR00950); and the Ministerio de Educacion y Ciencia (TIN2004-07739-C02-01). The authors thank Hillery Hunter from IBM, Gabe Loh from Georgia Tech., and anonymous reviewers for their detailed comments and suggestions.

References

[1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *International Conference on Computer-Aided Design*, November 2003.

[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 Transactions on Very Large Scale Integration Systems*, 13(1), January 2005.

[3] A. J. Bhavnagarwala, X. Tang, and J. D. Meindl. The impact of intrinsic device fluctuations on CMOS SRAM cell stability. *IEEE Journal of Solid-State Circuits*, 36(4), April 2001.

[4] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variation and impact on circuits and microarchitecture. In *40th Design Automation Conference*, June 2003.

[5] K. Bowman, S. Duvall, and J. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Journal of Solid-State Circuits*, 37(2), February 2002.

[6] D. Burger, J. Goodman, and A. Kagi. The declining effectiveness of dynamic caching for general-purpose microprocessors. Technical Report TR-1216, U.W.-Madison, Computer Science, 1995.

[7] B. Cline, K. Chopra, and D. Blaauw. Analysis and modeling of CD variation for statistical static timing. In *International Conference on Computer-Aided Design*, November 2006.

[8] R. Desikan, D. Burger, S. Keckler, and T. Austin. Sim-Alpha: a validated, execution-driven Alpha 21264 simulator. In *TR-01-23, CS Department, University of Texas*, 2001.

[9] P. Friedberg, W. Cheung, and C. J. Spanos. Spatial variability of critical dimensions. In *Proceedings of VLSI/ULSI Multilevel Interconnection Conference*, 2005.

[10] J. Friedrich et al. Design of the Power6 microprocessor. In *International Solid-State Circuits Conference*, Feb 2007.

[11] E. Humenay, D. Tarjan, W. Huang, and K. Skadron. Impact of parameter variations on multicore architectures. In *Workshop on Architectural Support for Gigascale Integration (ASGI-06, held in conjuction with ISCA-33)*, 2006.

[12] P. Juang, K. Skadron, M. Martonosi, Z. Hu, D. W. Clark, P. W. Diodato, and S. Kaxiras. Implementing branch predictor decay using quasi-static memory cells. *IEEE Transactions on Architecture and Code Optimization*, June 2004.

[13] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the International Symposium on Computer Architecture*, 2001.

[14] M. Khellah, Y. Ye, N. S. Kim, D. Somasekhar, G. Pandya, A. Farhang, K. Zhang, C. Webb, and V. De. Wordline and bitline pulsing schemes for improving SRAM cell stability in low-Vcc 65nm CMOS designs. In *2006 Symposium on VLSI Technology and Circuits*, June 2006.

[15] N. S. Kim, T. Kgil, K. Bowman, V. De, and T. Mudge. Total power-optimal pipelining and parallel processing under process variations in nanometer technology. In *International Conference on Computer-Aided Design*, November 2005.

[16] D. Lee and R. Katz. Using cache mechanisms to exploit nonrefreshing DRAMs for on-chip memories. *Journal of Solid-State Circuits*, 26(4):657–661, April 1991.

[17] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *39th IEEE International Symposium on Microarchitecture*, December 2006.

[18] W. K. Luk, J. Cai, R. H. Dennard, M. J. Immediato, and S. V. Kosonocky. A 3-transistor DRAM cell with gated diode for enhanced speed and retention time. In *2006 Symposium on VLSI Technology and Circuits*, June 2006.

[19] W. K. Luk and R. H. Dennard. A novel dynamic memory cell with internal voltage gain. *Journal of Solid-State Circuits*, 40(4), April 2005.

[20] D. Marculescu and E. Talpes. Variability and energy awareness: A microarchitecture-level perspective. In *DAC-42*, June 2005.

[21] K. Meng and R. Joseph. Process variation aware cache leakage management. In *International Symposium on Low Power Electronics and Design*, October 2006.

[22] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou. Yield-aware cache architectures. In *39th IEEE International Symposium on Microarchitecture*, December 2006.

[23] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick. A case for intelligent RAM. *IEEE Micro*, 17(2):34–44, 1997.

[24] A. Phansalkar, A. Joshi, L. Eeckhout, and L. K. John. Measuring program similarity: Experiments with SPEC CPU benchmark suites. In *IEEE International Symposium on Performance Analysis of Systems and Software*, March 2005.

[25] H. Pilo, J. Barwin, G. Braceras, C. Browning, S. Burns, J. Gabric, S. Lamphier, M. Miller, A. Roberts, and F. Towler. An SRAM design in 65nm and 45nm technology nodes featuring read and write-assist circuits to expand operating voltage. In *2006 Symposium on VLSI Technology and Circuits*, June 2006.

[26] B. F. Romanescu, S. Ozev, and D. J. Sorin. Quantifying the impact of process variability on microprocessor behavior. In *2nd Workshop on Architectural Reliability*, 2006.

[27] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002.

[28] D. Somasekhar, S.-L. Lu, B. Bloechel, K. Lai, S. Borkar, and V. De. A 10Mbit, 15GBytes/sec bandwidth 1T DRAM chip with planar MOS storage capacitor in an unmodified 150nm logic process for high density on-chip memory applications. In *31st European Solid-State Circuits Conference*, September 2005.

[29] A. Tiwari, S. R. Sarangi, and J. Torrellas. Recycle: Pipeline adaptation to tolerate process variation. In *Proceedings of the International Symposium on Computer Architecture*, 2007.

[30] R. K. Venkatesan, S. Herr, and E. Rotenberg. Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM. In *12th International Symposium on High-Performance Computer Architecture*, February 2006.

[31] D. A. Wood, M. D. Hill, and R. E. Kessler. A model for estimating trace-sample miss ratios. In *ACM SIGMETRICS*, June 1991.

[32] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *IEEE International Symposium on Quality Electronic Design*, 2006.