Trading off Cache Capacity for Reliability to Enable Low Voltage Operation

Chris Wilkerson, Hongliang Gao*, Alaa R. Alameldeen, Zeshan Chishti,
Muhammad Khellah and Shih-Lien Lu
Microprocessor Technology Lab, Intel Corporation **University of Central Florida*

Abstract

One of the most effective techniques to reduce a processor's power consumption is to reduce supply voltage. However, reducing voltage in the context of manufacturing-induced parameter variations can cause many types of memory circuits to fail. As a result, voltage scaling is limited by a minimum voltage, often called Vccmin, beyond which circuits may not operate reliably. Large memory structures (e.g., caches) typically set Vccmin for the whole processor.

In this paper, we propose two architectural techniques that enable microprocessor caches (L1 and L2), to operate at low voltages despite very high memory cell failure rates. The Word-disable scheme combines two consecutive cache lines, to form a single cache line where only non-failing words are used. The Bit-fix scheme uses a quarter of the ways in a cache set to store positions and fix bits for failing bits in other ways of the set. During high voltage operation, both schemes allow use of the entire cache. During low voltage operation, they sacrifice cache capacity by 50% and 25%, respectively, to reduce Vccmin below 500mV. Compared to current designs with a Vccmin of 825mV, our schemes enable a 40% voltage reduction, which reduces power by 85% and energy per instruction (EPI) by 53%.

1. Introduction

Ongoing technology improvements and feature size reduction have led to an increase in manufacturing-induced parameter variations. These variations affect various memory cell circuits including Small Signal Arrays (SSAs) (e.g., SRAM cells), and Large Signal Arrays (LSAs) (e.g., register file cells), making them unreliable at low voltages. The minimum voltage at which these circuits can reliably operate is often referred to as Vccmin.

Modern microprocessors contain a number of large memory structures. For each of these structures, the bit with the highest Vccmin determines the Vccmin of the structure as a whole. Since defective cells are distributed randomly throughout the microprocessor, the memory structures with the highest capacity (the L1 data and instruction caches and the L2 cache) typically determine the Vccmin of the microprocessor as a whole.

Vccmin is a critical parameter that prevents the voltage scaling of a design. Voltage scaling is one of the most effective ways to reduce the power consumed by a microprocessor since dynamic power is a quadratic function of Vcc and leakage is an exponential function of Vcc [17][1]. Therefore, Vccmin is a critical parameter that prevents reducing a particular design's power consumption. Overcoming Vccmin limits allows designs to operate at lower voltages, improving energy consumption and battery life for handheld and laptop products.

This paper shows how reducing Vcc below Vccmin affects the reliability of a microprocessor. Based on published bit failure data, we estimate that a 2 MB cache implemented in 65 nm technology (similar to L2 size of the Intel® Core™ 2 Duo processor) has a Vccmin of 825 mV. We examine two novel architectural mechanisms to enable some of the largest memory structures (specifically the L1 data and instruction caches and the second level cache) to be redesigned to decrease Vccmin. These schemes identify and disable defective portions of the cache at two different granularities: individual words or pairs of bits. With minimal overhead, both schemes significantly reduce the Vccmin of a cache in low-voltage mode while reducing performance

[1]Leakage has an exponential dependence on device threshold voltage (Vth) and Vth in turn is a function of Vcc as a result of the Drain Induced Barrier Lowering (DIBL) effect [17].

marginally in high-voltage mode. Both schemes achieve a significantly lower overhead compared to ECC-based defect tolerance schemes at low voltages.

In the first scheme, Word-disable, we disable 32-bit words that contain defective bits. Physical lines in two consecutive cache ways combine to form one logical line where only non-failing words are used. This cuts both the cache size and associativity in half in low-voltage mode. Each line's tag includes a defect map (one bit per word, or 16 bits per 64-byte cache line) that represents which words are defective (0) or valid (1). This scheme allows a 32KB cache to operate at a voltage level of 490mV.

In the second scheme, Bit-fix, we use a quarter of the cache ways to store the location of defective bits in other ways, as well as the correct value of these bits. For an 8-way cache, for example, we reserve two ways (in low-voltage mode) to store defect-correction information. This reduces both the cache size and associativity by 25% in low-voltage mode, while permitting a 2MB cache to operate at 475 mV.

In this paper, we make the following main contributions:

1. We highlight the importance of Vccmin and its impact on power and reliability.
2. We propose two schemes to enable large memory structures to operate at low-voltages. Both schemes achieve reliable operation of caches at 500mV with minimal overhead, while reducing performance marginally when the processor operates at the normal voltage level.
3. To our knowledge, this is the first paper to show architectural techniques that allow reliable cache operation from unreliable components at sub-500mV voltages.
4. We demonstrate that our schemes can reduce overall microprocessor power significantly compared to a system with traditional caches.

In the remainder of this paper, we discuss the importance of Vccmin and its impact on power (Section 2). We introduce our model for failure probability in Section 3. We discuss some prior work in Section 4. We then discuss our two proposed schemes in detail in Section 5. We introduce our experimental methodology in Section 6. Section 7 evaluates our proposed schemes in terms of performance, power, and area overhead, and we conclude in Section 8.

2. Vccmin and its impact on cache power and reliability

A conventional six-transistor (or 6T) SRAM cell is shown in Figure 1 [13]. The cell consists of two identical CMOS inverters, connected in a positive feedback loop to create a bi-stable circuit allowing the storage of either "0" or "1", and two NMOS access transistors. A word-line select signal (WL) is applied to the access transistors to allow reading or writing the cell by connecting the bit-lines (BL0, BL1) to the cell storage nodes. The SRAM cell is symmetrical and is divided into a left side and a right side. In Figure 1 the left side is assumed to be initially storing a "0" while the right side is assumed initially storing a "1".

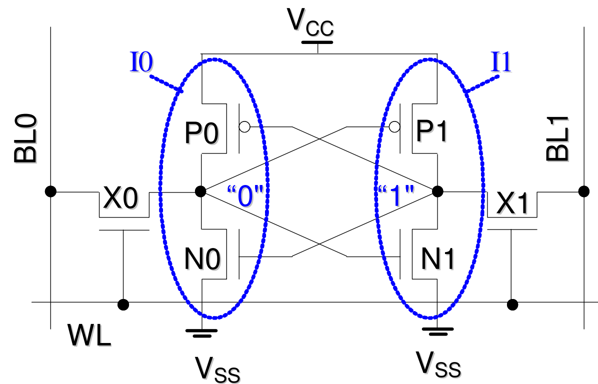

Figure 1. Conventional six-transistor (6T) SRAM cell

To read, bit-lines are pre-charged to Vcc equally first. The word-line select pulse is then applied to turn on the access transistors allowing a differential voltage (50-100mV) to build across the bit-lines. After that, a sense-amplifier connected to the bit-lines is activated to amplify the bit-line voltage differential to provide the read value. To write, bit-lines are driven to the desired value first. The word-line select pulse is then applied allowing write value to drive into the cell through access transistors.

2.1 Types of cell failures

An SRAM cell can fail in four different ways, which we describe next [2, 11, 16]:

Read failure. A read failure occurs when the stored value is flipped during a read operation. This happens when the noise developed on the node storing "0" (due to charge sharing with the highly-capacitive pre-charged BL0) is larger than the trip point of inverter I1.

Write failure. A write failure occurs when the cell contents can't be toggled during a write operation. To toggle the contents of the cell in Figure 1, BL1 is driven to Vss while BL0 remains pre-charged to Vcc. In an operational cell, transistor X1 will start discharging node I1 (on the right) and positive feedback will complete the operation driving I0 (on the left) to 1. Strengthening the pull-up device P1 vs the access device X1 will make toggling the contents of the cell harder and cause write failures. Write and read failures dominate the other two types of cell failures.

Access failure. An access failure happens when the differential voltage developed across the bit-lines during a read operation is not sufficient for the sense-amplifier to identify the correct value. Increasing the WL pulse period usually helps reduce access failure.

Retention (hold) failure. A retention failure occurs when the stored value in the cell is lost during standby. This could be the result of a voltage drop for example.

2.2 Impact of supply voltage on reliability

At high supply voltages, cell operating margins are large, leading to reliable operation. For example, during a read operation, the voltage bump induced on the "0" side of the cell is much smaller than the noise margin of inverter I1, which prevents the cell from losing its contents. However, reducing supply voltage decreases cell noise margins. This decrease, coupled with parametric variation due to manufacturing imperfections, significantly limits the minimum supply voltage (Vccmin) below which the SRAM cell can no longer operate. In particular, intra-die Random Dopant Fluctuations (or RDF) play a primary role in cell failure by arbitrarily impacting the number and location of dopant atoms in transistors, resulting in different Vth for supposedly matched SRAM devices [1]. For example, RDF can make X0 stronger than N0 resulting in a read unstable cell, or making X1 weaker than P1 resulting in write failures for another cell.

Random intra-die variations (such as RDF) are modeled as random variables and used as inputs to determine failure probability (or Pfail) of a given SRAM cell [1, 2]. Figure 2 shows an example cell failure probability as a function of supply voltage Vcc [8][2]. As Vcc decreases, the failure probability increases exponentially. This increase can be attributed to the higher sensitivity of cell stability to parametric variations at lower supply voltages. Thus, stability failures impose a limit on the minimum supply voltage.

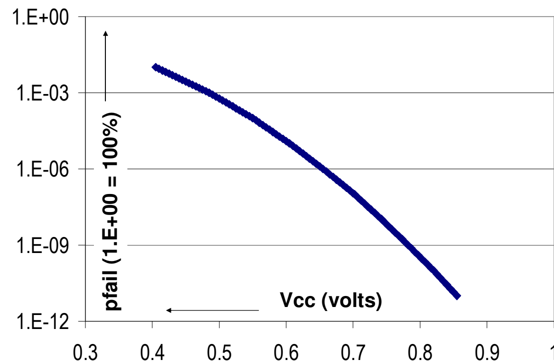

Figure 2. Probability of failure (Pfail) for a cell vs. Vcc [8]

3. Model for failure probability (Pfail)

As suggested in the previous section, we assume that stability failures are randomly distributed throughout a memory array. Each cell has a probability of failure (Pfail). A die containing even a single cell failure must be discarded. The aggregate probability of failure for all memory arrays on the die is a significant component of overall manufacturing yield. In our analysis, we assume that the Pfail for each memory array must be kept at less than 1 out of 1000 to achieve acceptable manufacturing yields. With these assumptions, we derive nominal Vccmin as the voltage at which every cell is functional in 999 out of every 1000 dies. Figure 3 shows the Vccmin values for a 2MB and a 32KB cache (representative of the L2 and L1 I&D caches for the Intel® Core™ 2 Duo processor [4]). Achieving acceptable manufacturing yields for a 2 MB cache requires a Vccmin of nearly 825 mV. Because of its smaller size, the 32KB cache has a lower Pfail than that for the 2MB cache. However, one cannot decrease Vcc below nearly 760 mV for the 32KB cache without sacrificing yield.

One possible mechanism to decrease cache Vccmin is to selectively disable defective data in the cache. Such disabling can be carried out at

---

[2] Vccmin is closely related to the yield of a product. As such, industrial Vccmin data is unavailable for publication. Data on cell failure rates for different SRAM designs are based on simulated/measured results reported in [8]. The analysis in [8] focuses on a 130nm process node. This paper focuses on a 65nm technology but assumes a similar Pfail/Vcc slope. This assumes that variations (RDF in particular) won't get worse on the newer technology node. Worsening variations will make Vccmin worse and increase the need for aggressive mitigation mechanisms such as those discussed in this paper.

different granularities, ranging from cache ways [12], entire cache lines [3] (coarse), to individual bits (fine). To explore such possibilities, we show the relationship between Vcc and Pfail for a 64-byte line, a byte, and a bit in Figure 3. While the failure probability for a 64B line is less than that for the whole cache, almost every cache line has at least one defective bit at Vcc below 500 mV. Because a cache line consists of many cells and the failure of any one of these cells could render the cache line defective, the Pfail of a cache line is significantly higher than that of individual bytes and bits. The data in Figure 3 suggests that disabling at finer granularities could enable operation at Vcc values below 500 mV. We explore mechanisms for finer granularity disabling in detail in Section 5.

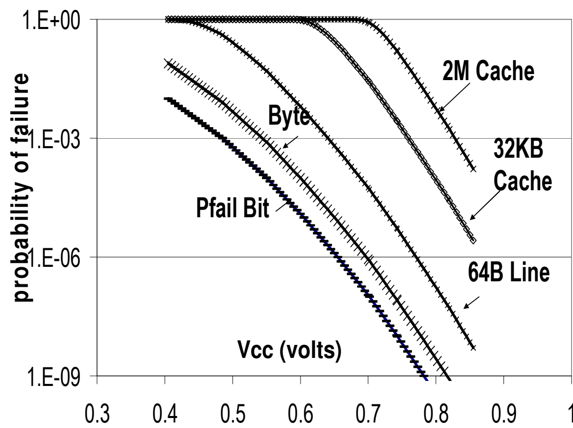

Figure 3. Pfail vs. Vcc for various cache structures

In addition to parametric failures, several other types of failures may occur in an SRAM cell. For example, hard failures (due to short or open defects) are typically mitigated through column/row redundancy. Soft errors, failures due to device aging such as Negative Bias Temperature Instability (NBTI) [9], and erratic bit behavior due to gate oxide leakage [10] effect FIT (Failures in Time) rates. These can be addressed with ECC and additional supply voltage (Vcc) guardband.

4. Related work

Khellah, et al., propose a Dual-Vcc approach to allow the core to operate at a low voltage while maintaining a separate power supply for large memory structures [6]. Although this approach increases platform cost, it may be practical for large memory structures that are segregated from the core (e.g., L2 cache). For memory structures embedded in the core (e.g., L1 I&D caches), differing adjacent voltage domains add a great deal of design complexity. Voltage-level converters must be added to allow signal sharing between the memory structures running at high voltages and logic blocks running at low voltages. Furthermore, accesses to memory structures running at high voltages generate noise on nearby low voltage blocks requiring either shielding or additional noise margin. The most serious drawback for this approach is that power consumption in the high voltage structures will quickly begin to dominate total power consumption as aggressive voltage scaling reduces power consumption on the rest of the die. As an example, the L1 caches might account for 20% of the dynamic power consumption at high voltage. After a voltage reduction of 40%, classic voltage scaling for the rest of the die would indicate that the L1 caches account for 55% of the dynamic power. In our analysis, we assume that our platform provides a single power supply, and Vccmin in the large memory structures determines Vccmin for the die.

A cache designer has several alternatives to improve the reliability of memory cells at low voltages. The least intrusive option addresses the problem by changing the basic design of the cell. The designer can upsize the devices in the cell, reducing the impact of variations on device performance and improving the stability of the cell. Variations on the traditional 6-T (6 transistor) SRAM design can also be adopted including 8-T and 10-T cells. Kulkarni et al. [3] compare some of these approaches with their proposed Schmidt Trigger (ST) 10-T SRAM cell. For a fixed area, they find that the ST cell delivers better low voltage reliability than 6-T, 8-T, and 10-T SRAM cell designs. The improved reliability of the ST cell comes at the cost of a 100% area increase and an increase in latency. Figure 4 shows the Vccmin reduction achievable if we use the ST cell for a 2MB cache. The ST cell allows us to reduce Vccmin to 530mV for the 2MB cache. As we show in Section 5, our proposals achieve lower Vccmin than the ST cell with less area.

A simple solution to increase reliability, which has been used by industry, is implementing row and column redundancy in caches [14]. In these schemes, one or several additional rows and/or columns are added to a cache array design, and these additional cells are enabled when other cache cells fail. However, these techniques are

impractical for dealing with the very high failure rates we consider. While row/column redundancy is effective for dealing with tens of defective bits, our techniques allow us to address caches with thousands of defective bits.

Instead of improving the design of the cell, a designer can choose to build error detection/ correction into the array by choosing from a wide variety of available error detection/correction codes (EDC/ECC). ECC has proven to be an attractive option for reducing cache vulnerability to transient faults such as soft errors where multi-bit errors are unlikely. Figure 4 shows that 1-Bit ECC or SECDED (single-error correction, double-error detection) can reduce Vccmin by around 150mV. However, SECDED falls far short of our 500mV goal. To allow operation around 500mV, we would need to augment our 2MB cache with a 10-bit ECC. However, multi-bit error correction codes have high overhead [7], requiring storage for correction codes as well as complex and slow decoders to identify the errors. As an example, Kim, et al. [7], propose a scheme to use 2-D ECC to detect and correct multi-bit errors. While this scheme requires less overhead in check bits, it requires a read-modify-write operation for every write and many extra cycles (depending on the organization) to correct. Hsiao, et. al. [5], propose a class of one-step majority-decodable codes called Orthogonal Latin Square Codes (OLSC) that is both fast and implementable for multi-bit error correction. Encoding check bits for t-bit errors (10 in our case) and m^2 data bits (m^2 = 512 in our case) requires 2*t*m (m-1)-input XOR gates. To decode and correct the transmitted data, we need $2tm^2$ m-input XOR gates and a (2t+1)-input majority function gate [3] for each data bit. Unfortunately, this class of codes have substantial storage overhead, since m^2 data bits require 2*t*m check bits. The number of check bits grows $O(m^{1/2})$ rather than the theoretic limit of ECC of O(log n). Implementing our 10-bit ECC with OLSC to allow operation at 500mV would require 460 check bits for a 512 bit cache line. Therefore, a 10-bit ECC introduces a significant (nearly 2X) area overhead, as well as significant complexity to correct errors. In the next section, we describe our two proposals to achieve lower Vccmin at a much lower overhead.

3 A majority function gate of n-input outputs a '1' if the majority of the n inputs is '1'.

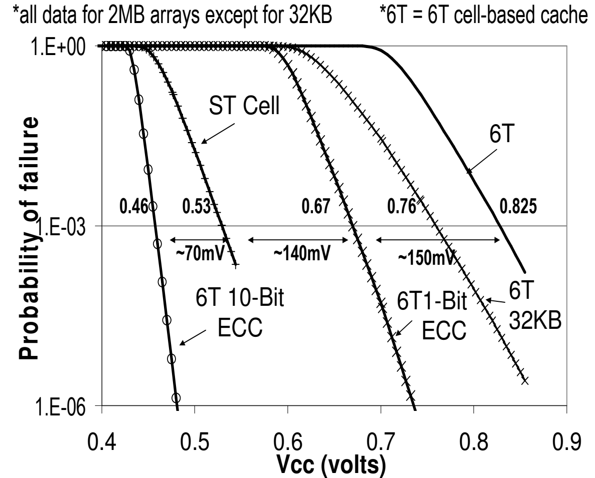

Figure 4. Pfail for different failure mitigation schemes

5. Two defect-tolerance mechanisms

Ideally, a mechanism to handle defective bits should be turned off in high-performance mode when high voltages preclude defects, and turned on at low voltages when bit failures are pervasive. In this section, we describe two novel architectural mechanisms that achieve this objective. Both mechanisms trade off cache capacity at low voltages, where performance (and cache capacity) may be less important, to gain the improved reliability required for low voltage operation. At high voltages where performance is critical, both mechanisms have minimal overhead.

To achieve this, we leverage low voltage memory tests performed when the processor boots [3]. These memory tests identify parts of the cache that will fail during low voltage operation allowing defect avoidance and repair. The first mechanism, Word-disable identifies and disables defective words in the cache. The second mechanism, Bit-fix, identifies broken bit-pairs and maintains patches to repair the cache line. Both mechanisms address bit failures in the cache data array but not the tag array. As a result, we implement the tag arrays of our caches using the ST cell, which allows the 32KB and 2MB cache tag arrays to operate at 460mV and 500mV respectively. The tag array for the 2MB cache can also be implemented with SECDED ECC, which will allow correct operation at voltages as low as 400mV.

5.1 Cache word-disable

The word-disable mechanism isolates defects on a word-level granularity and then disables words containing defective bits. Each cache line's tag keeps a defect map with one bit per word that represents whether the word is defective (1) or valid (0). For each cache set, physical lines in two consecutive ways combine to form one logical line. After disabling defective words, the two physical lines together store the contents of one logical line. This cuts both the cache size and associativity in half.

Example. Assume we have a 32KB 8-way L1 cache containing 64-byte lines. For each cache set, eight physical lines correspond to four logical lines. We use a fixed mapping from physical lines to logical lines: Lines in physical ways 0 and 1 combine to form logical line 0, lines in physical ways 2 and 3 combine to form logical line 1, and so on. The first physical line in a pair stores the first eight valid words, and the second stores the next eight valid words of the logical line for a total of 16 32-bit words. As a result, in low-voltage mode, the cache effectively becomes a 16KB 4-way set associative cache. In the tag array, each tag entry is required to store a 16-bit defect map that consists of one defect bit for each of the 16 32-bit words. These bits are initialized at system boot time and are kept unchanged in both high and low voltage modes.

Operation in low-voltage mode. During the tag match, the way comparators try to match the incoming address's tag bits with the address tag for even ways 0, 2, …etc. We do not need to match tags in odd ways since they contain the same information as their pair partners. If one of the tags matches, we select the even way (e.g., way 0) or the odd way based on the sixth-least significant bit of the address tag, since half the words are stored in the even/odd way. Requests that require data from 2 separate 32-byte lines are treated as accesses that cross a cache line boundary.

To obtain the 32-byte data in aligned form, we use two four-stage shifters to remove the defective words and aggregate working words as shown in Figure 5. We divide the cache-line into two halves, each with a maximum of four defective words, and each storing four of the eight required word. A line with more than four defective words in either half renders the whole cache defective. Each of the shifting stages depicted in Figure 5 "shifts out" a single defective word. To control each shifter we decompose the defect-map into four separate bit-vectors, each a 1-hot representation of the location of a different defective word.

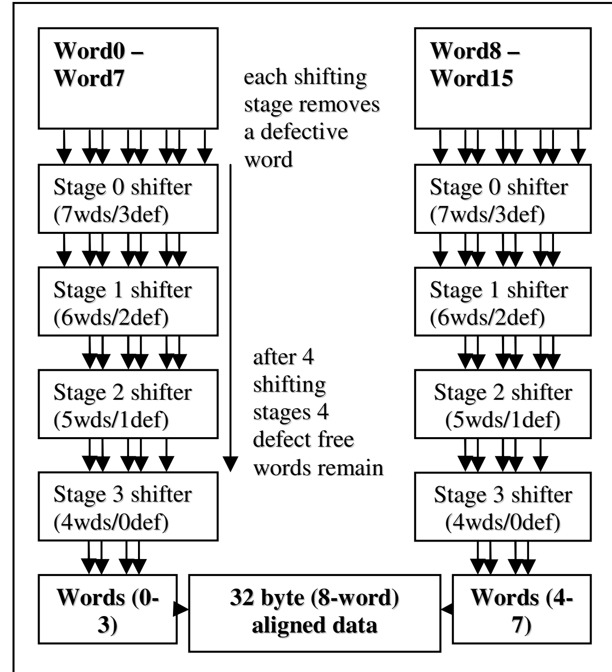

Figure 5. Overview of cache word-disable operation

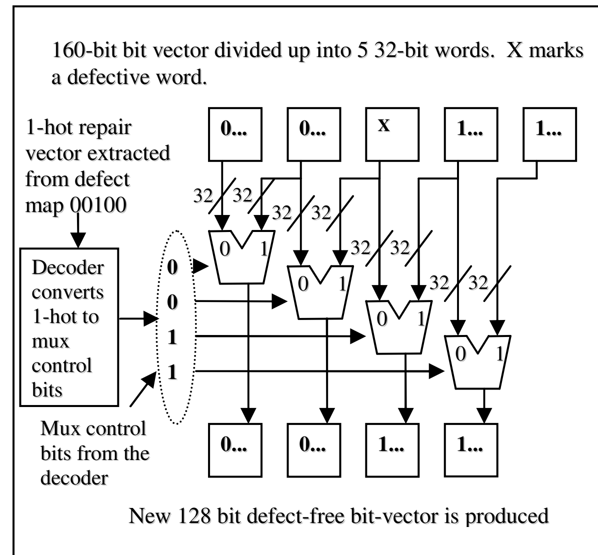

Figure 6. Word-disable operation: removing a defective word

Figure 6 shows a simplified version of the logic used to disable and remove a single defective word from a group of words. Starting with the defect map, we extract a 1-hot repair vector identifying the position of a single defective word. In the figure, the vector "00100" identifies the 3rd word as defective. The

decoder converts the 1-hot vector into a Mux-control vector containing a string of 0s up to (but not including) the defective position followed by a string of 1s. This has no effect on the words to the left of the defect, but each of the words to the right of the defective word shifts left, thereby "shifting-out" the defective word. Each level of 32-bit 2:1 muxes eliminates a single defective word. Our proposed implementation must eliminate four defective words, requiring a cache line to pass through four levels of muxes. The additional multiplexing, and associated logic for Mux-control, will add latency to the cache access. Assuming 20 FO4 delays per pipeline stage, we estimate that the additional logic increases the cache latency by one cycle.

5.2 Cache bit-fix

The Bit-fix mechanism differs from the Word-disable mechanism in three respects. First, instead of disabling at word-level granularity, the Bit-fix mechanism allows groups of 2-bits to be disabled. These defective pairs are groups of 2-bits in which at least one bit is defective. Second, for each defective pair, the Bit-fix mechanism maintains a 2-bit patch that can be used to correct the defective pair. Third, the Bit-fix mechanism requires no additional storage for repair patterns, instead storing repair patterns in selected cache lines in the data array. This eliminates the need for the additional tag bits required to store the defect map in the word-disable mechanism, but introduces the need to save repair pointers elsewhere in the system (e.g., main memory) while they are not in use (in high-voltage mode).

During low-voltage operation, the repair patterns (repair pointers as well as patches) are stored in the cache. As a result, any read or write operation on a cache-line must first fetch the repair patterns for the cache line. When reading, repair pointers allow reads to avoid reading data from broken bits. Using patches from the repair patterns, the cache line is reconstructed before being forwarded to the core, another cache, or written back to memory. When writing, repair pointers allow writes to avoid writing to broken bits. New patches must be written to the repair patterns to reflect new data written to the cache. Although we focus on reads in the following discussion, reads and writes are symmetric.

Example. Assume we have a 32KB 8-way L1 cache containing 64-byte lines. Each access to data stored in the cache requires an additional access to retrieve the appropriate repair patterns. To access the repair patterns without increasing the number of ports, the Bit-fix scheme organizes the cache into two banks. For our 8-way cache we maintain two fix-lines, one in each bank. As we show later in this section, the repair patterns for three cache lines fit in a single cache line. Therefore, we maintain a single fix-line (a cache line storing repair patterns) for every three cache lines. A fix-line is assigned to the bank opposite to the three cache lines that use its repair patterns. This strategy allows a cache line to be fetched in parallel with its repair patterns without increasing the number of cache ports.

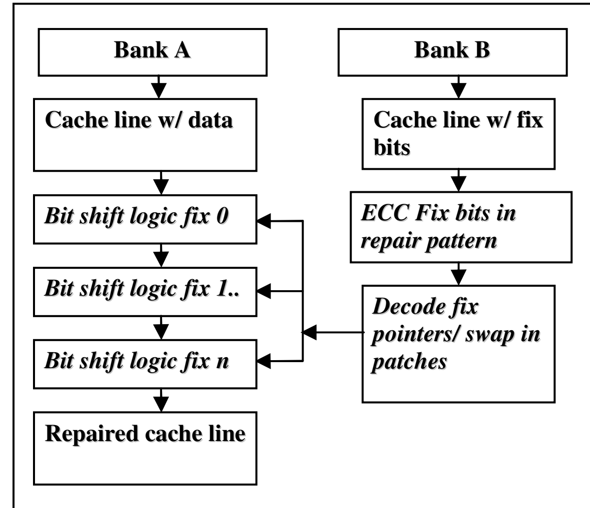

Figure 7. High-level operation of the bit-fix scheme

Operation in low-voltage mode. Figure 7 shows the operation of the Bit-fix mechanism at a high level. On a cache hit, both the data line and fix-line are read. In this figure, we fetch the data line from Bank A and the fix-line from Bank B. The data line passes through 'n' bit shift stages, where 'n' represents the number of defective bit pairs. Each stage removes a defective pair, replacing it with the fixed pair. Since the fix-line may also contain broken bits, we apply SECDED ECC to correct the repair patterns in the fix-line before they are used. After the repair patterns have been fixed, they are used to correct the data-line. Repairing a single defective pair consists of three parts. First, SECDED ECC repairs any defective bits in the repair pattern. Second, a defect pointer identifies the defective pair. Third, after the defective pair has been removed, a patch reintroduces the missing correct bits into the cache line.

We demonstrate how Bit-fix works for a short 10-bit line in Figure 8. The logic depicted in Figure 8 can be generalized to 512-bit lines. The 10-bit line consists of five 2-bit segments. We indicate the defective bit with an X and the valid bits as 0 or 1. The defective

pair contains “X1”. To repair the line, we first identify and remove the defective pair. For example, the defect pointer “000” indicates that the first 2-bit pair is defective. In Figure 8, the defect pointer “010” indicates that the third 2-bit pair is defective. We decode the defect pointer into a 5 bit Mux-control vector in a similar fashion to the Word-disable scheme, where the defect position and all the bits to its right are set to 1, and all other bits are set to 0. In Figure 8, a defect in the third 2-bit pair produces the control vector “00111”. The Mux-control is used to control multiplexers that create a new “repaired” line by shifting all bits in the original line located to the right of the defective pair two positions to the left, while keeping the other bits unchanged, thereby “shifting-out” the defective pair. We restore the vector to its original state by appending the patch “01”, where “0” is the correct value of the defect bit “X”.

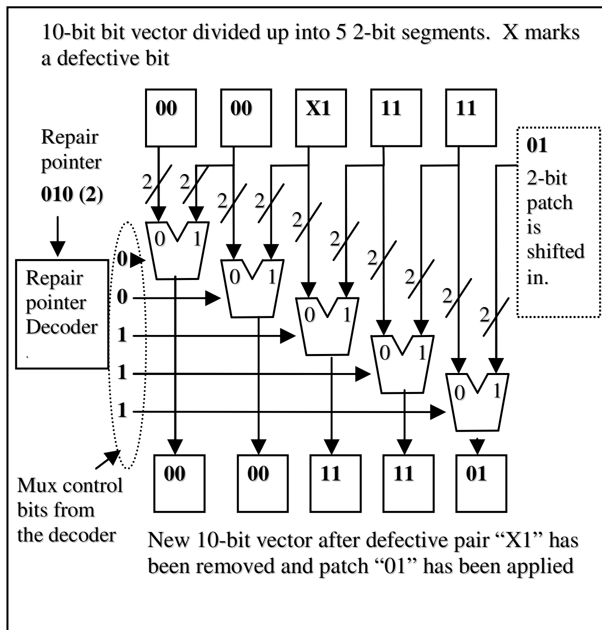

Figure 8. Implementation of the bit-fix scheme for ten bits

For 512-bit lines, Bit-fix organizes the line as 256 pairs that require 256 2:1 2-bit multiplexers arranged similar to Figure 8. To repair ten defective pairs, we require ten repair stages, each with a separate decoder. Using synthesis tools we have found that each decoder requires 306 FO4 gates with three inputs or less. In total, the logic required to implement this function is 2560 2:1 2-bit muxes and 3060 FO4 gates with a maximum of three inputs, for a total of fewer than 26,000 transistors. Due to the need for ECC on repair lines and the large number of repair stages, we estimate that repairing up to ten defective bits would increase cache access latency by three cycles, assuming 20 FO4 delays per cycle.

Fix Line Structure. To repair a single defective pair in a 512-bit cache line, each defect pointer requires 8 bits ($\log_2 256 = 8$), and each patch requires 2 bits, in addition to four bits of SECDED ECC to compensate for broken bits in our repair pattern. In total, we need 14 bits for a single defect. Our Pfail model estimates that, with a bit failure probability of 0.001, three out of every billion cache lines contain more than ten defective bit pairs. A 140-bit repair pattern can effectively repair a single data cache line with ten or fewer defects. Each 512-bit fix line will store three 140-bit repair patterns, for a total of 420 bits.

5.3 Discussion

Vccmin improvements. We demonstrate the failure probabilities of our proposed schemes in Figures 9 and 10. For a failure probability of 0.001, a 32 KB cache can reliably operate at 490mV, and a 2 MB cache can reliably operate at 510mV using the Word-disable scheme. For Bit-fix, a 32KB cache can reliably operate at 455mV, and a 2MB cache can reliably operate at 475 mV. Compared to a modern high-performance microprocessor, where low-power voltages are in the range of 800-850mV, this represents more than a 40% drop in Vccmin. We show this significantly reduces power in the results section.

Figures 9 and 10 also compare the Vccmin for Word-disable and Bit-fix schemes with conventional 6T cell, 6T cell with 1-bit ECC, and the previously proposed ST cell-based cache. Both Word-disable and Bit-fix show significant Vccmin reduction in comparison with the 6T cell and 6T cell with 1-bit ECC. For a 2MB cache, Word-disable and Bit-fix decrease Vccmin by 20 mV and 55 mV respectively without incurring the cell area overhead of the much larger ST cell.

Comparison. Both the Word-disable and Bit-fix schemes require memory tests to be performed when the processor is powered on [3]. When switching between low-voltage and high-voltage modes, the cache contents need to be flushed to memory. In addition, both schemes require that the reliability of the tags be improved through the use of ST cells and potentially ECC. The size of the tag array is roughly 5% of the size of the data array. The use of the ST cells and ECC will increase the size of the tag array by roughly 2.5X for the Bit-fix mechanism, and 4X for the word disable mechanism which also requires a more reliable defect map. For a 2MB cache, the Bit-

fix mechanism increases the cache size by ~7%, and the Word-disable mechanism increases the cache size by ~15%. Compared to Word-disable, Bit-fix has a lower area overhead, operates at a lower Vccmin, and preserves three-fourths of the cache for use at low voltages vs. one-half for the Word-disable approach. However, Word-disable requires less complex muxing and as a result has a smaller cache latency. Word-disable's tag overhead includes storage for the defect map, which eliminates the Bit-Fix complexity of using the cache to store repair patterns.

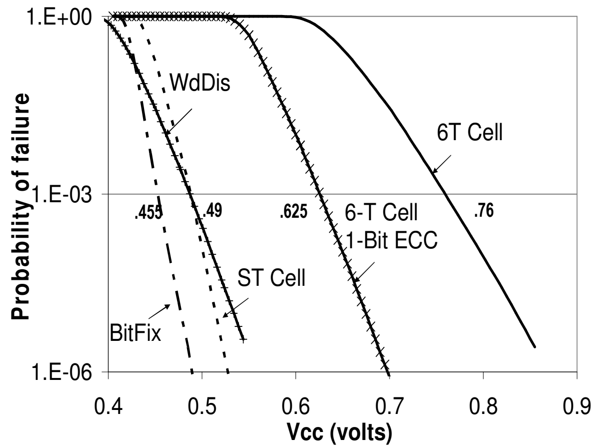

Figure 9. Pfail of 32KB cache with different failure mitigation schemes

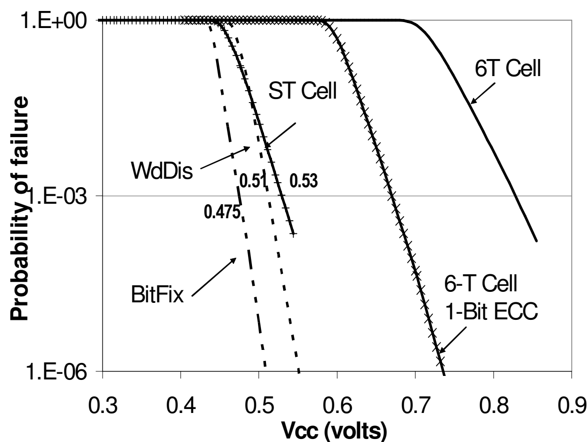

Figure 10. Pfail of 2MB cache with different failure mitigation schemes

6. Simulation methodology

We use a cycle-accurate, execution-driven simulator running IA32 binaries. The simulator is micro-operation (uOp) based, executes both user and kernel instructions, and models a detailed memory subsystem. Table 1 shows the parameters of our baseline out-of-order processor, which is similar to the Intel® Core™ 2 Duo processor on 65nm technology [4]. In order to quantify the relative performance loss as a result of cache latency and capacity changes introduced by our schemes, we also simulate a defect-free low-voltage baseline that has reliable caches without any latency overhead or capacity loss. Since the reduction of supply voltage causes transistor switching delay to increase, we perform circuit simulations to predict the frequencies at different voltages. In order to simplify our discussion, we fix the low voltage at 500mV when applying our schemes instead of using a different voltage for each scheme. In the following sections, we annotate the results of a processor that implements Scheme A for the L1 cache and Scheme B for the L2 cache as "L1A_L2B". For example, "L1WDis_L2BFix" means that the L1 cache uses the Word-disable scheme, while the L2 cache uses the Bit-fix scheme.

Table 1. Baseline processor parameters

Voltage Dependent Parameters		
	High Vol.	Low Vol.
Processor frequency	3 GHz	500 MHz
Memory latency	300 cycles	50 cycles
Voltage	1.3V	0.5V
Voltage Independent Parameters		
ROB size	256	
Register file	256 fp, 256 int	
Fetch/schedule/retire width	6/5/5	
Scheduling window size	32 FP, 32 Int, 32 Mem	
Memory disambiguation	Perfect	
Load buffer size	32	
Store buffer size	32	
Branch predictor	16k bytes TAGE [15]	
Hardware data prefetcher	Stream-based (16 streams)	
Cache line size	64 bytes	
L1 Instruction cache	32kB, 8-way, 3 cycles	
L1 Data cache	32kB, 8-way, 3 cycles	
L2 Unified cache	2MB, 8-way, 20 cycles	

In our experiments, we simulate nine categories of benchmarks. For each individual benchmark, we carefully select multiple sample traces that well represent the benchmark behavior. Table 2 lists the number of traces and example benchmarks included

in each category. We use instructions per cycle (IPC) as the performance metric. The IPC of each category is the geometric mean of IPC for all traces within that category. Then we normalize the IPC of each category to the baseline to show performance.

Table 2. Benchmark suites

Category	# of traces	Example benchmarks
Digital home (DH)	60	H264 decode/encode, flash
FSPEC2K (FP2K)	25	www.spec.org
ISPEC2K (INT2K)	26	www.spec.org
Games (GM)	49	Doom, quake
Multimedia (MM)	77	Photoshop, raytracer
Office (OF)	52	Excel, outlook
Productivity (PROD)	43	File compression, Winstone
Server (SERV)	48	SQL, TPCC
Workstation (WS)	82	CAD, bioinformatics
ALL	462	

7. Results

In this section, we present the experimental results of our proposed schemes. Section 7.1 evaluates the performance impact of the proposed schemes and discusses the tradeoffs involved in choosing an appropriate scheme at each cache level. Section 7.2 compares our schemes with the ST cell-based cache in terms of area overhead, power consumption, and energy efficiency.

7.1 Performance

In this section, we evaluate the performance overhead of our two proposals in both high-voltage and low-voltage modes. For both operating modes[4], Word-disable has a 1-cycle latency overhead and Bit-fix has a 3-cycle latency overhead (Section 5). We evaluate different combinations of our two schemes for the two cache levels. Recall from Section 5.3 that amongst the two schemes, Word-disable has a latency advantage while Bit-fix provides more capacity in the low-voltage mode.

[4] For the high-voltage mode, it is possible to design bypass networks to bypass the logic required for the low-voltage mode. However, we chose to evaluate using a conservative estimate for latencies in the high-voltage mode.

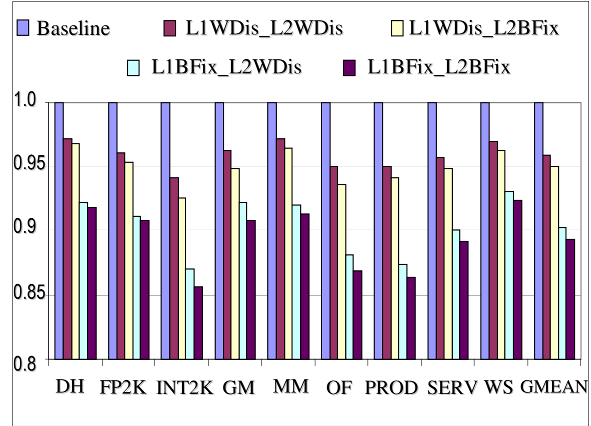

Figure 11. Normalized IPCs in the high-voltage mode

Figure 11 shows normalized IPC relative to the high-voltage baseline, when applying different combinations of Word-disable and Bit-fix schemes to the L1 and L2 caches in the high-voltage mode. Performance in high-voltage mode is sensitive to the L1 latency and more tolerant to increases in the L2 latency. When we use Word-disable for the L1 cache and Word-disable (Bit-fix) for the L2 cache, IPC reduces by 4% (5%). On the other hand, if we use Bit-fix for the L1 cache and Word-disable (Bit-fix) for the L2 cache, IPC reduces by 9.7% (10.5%). The INT2K, OF, SERV, and PROD benchmarks are very sensitive to L1 latency and see a 10% performance loss when using Bit-fix for the L1 cache. To minimize additional latency in the L1 cache, we use Word-disable for the L1 cache in the remainder of this section.

To quantify the performance overhead of our schemes in the low-voltage mode, Figure 12 shows normalized IPC relative to the defect-free low-voltage baseline, when applying Word-disable to the L1 cache and Word-disable or Bit-fix to the L2 cache. When we apply Word-disable to the L1 cache, the Word-disable and Bit-fix schemes for the L2 cache have a similar impact on IPC (10.2% and 10.7% reductions, respectively). Five of the nine benchmark categories have more than 10% IPC reductions. However, this performance overhead is in comparison to a defect-free baseline that has reliable caches with no area or latency overhead at such low voltages. Moreover, as the low-voltage mode is normally used when the processor load is low, the performance is not the primary concern.

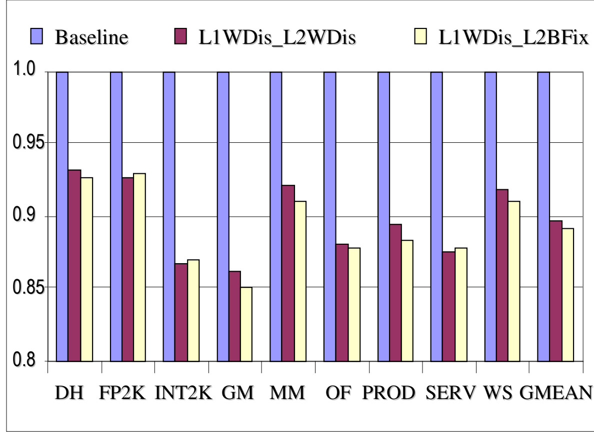

Figure 12. Normalized IPCs in the low-voltage mode

From the results shown in Figures 11 and 12, it is not clear which scheme is better for the L2 cache. To distinguish the two schemes, we examine the increases in memory accesses due to the capacity loss for both the schemes in Figure 13. Compared to the low-voltage baseline, Word-disable increases the number of memory accesses by 28% (up to 115%), while Bit-fix increases the number of memory accesses by only 7.5% (up to 13%). Bit-fix has fewer memory accesses due to its larger effective cache size and higher associativity compared to Word-disable. Since the extra memory accesses increase the platform power consumption, we argue that Bit-fix is a better choice than Word-disable for the L2 cache.

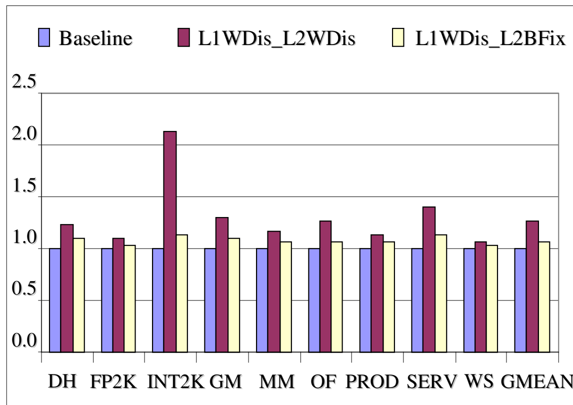

Figure 13. Normalized number of memory accesses

7.2 Comparison with ST cell-based cache

As shown in Section 4, the ST cell is more reliable than the conventional 6T cell and enables a Vccmin of 530 mV for a 2MB cache, which is close to the achievable Vccmin of our schemes. In this section we present detailed comparison between our schemes and the ST cell-based cache. Based on the analysis in Section 7.1, we use a processor that applies Word-disable to the L1 cache and Bit-fix to the L2 cache as a representative application of our schemes. Table 3 summarizes the achievable Vccmin, cache area, frequency, power consumption, IPC, and energy per instruction (EPI) of each scheme in the low-voltage mode. We normalize the area, power, and EPI results for each scheme to the corresponding results for the conventional 6T cell-based cache, while showing absolute results for Vccmin, frequency, and IPC. In order to estimate the power consumption, we assume that dynamic power scales quadratically with supply voltage, and linearly with frequency. We also assume that static power scales with the cube of supply voltage [17]. We give an artificial advantage to the ST cell-based cache in our power calculations by ignoring the leakage overhead due to ST cell's larger area. We also ignore the latency overhead of the ST cell in our performance simulations.

Table 3. Comparison between the ST cell-based cache and our proposed schemes

Scheme	Vccmin (mV)	Norm. cache area	Clock Speed (MHz)	Norm. Power	IPC	Norm. EPI
6T Cell	825	1	1900	1	1.02	1
ST Cell	530	2	700	0.19	1.17	0.45
L1WDis _L2BFix	500	1.08	600	0.15	1.07	0.47

Our schemes (L1WDis_L2BFix) achieve higher power reduction than the ST cell-based cache despite ignoring the ST cell's leakage power overhead. Compared to the processor with the 6T cell-based cache, our schemes reduce the power consumption by 85%, while the ST cell-based cache reduces the power consumption by 81%. Moreover, our schemes have significantly lower area overhead (8%) than the ST-cell based cache (100%). While one may reduce the area overhead of the ST cell-based cache by reducing the cache capacity, such reduction comes at the cost of sacrificing performance in the high-voltage mode.

Since we scale the memory latency with frequency in our performance simulations, the ST cell-based cache and our schemes have higher IPC

than the 6T cell-based cache. At the same time, lower frequency makes the execution time longer, which results in a smaller energy efficiency improvement than the power reduction. Although the latency overhead and capacity loss of our schemes in the low voltage mode result in a lower IPC compared to the ST cell-based cache, our schemes' lower power achieves a similar energy efficiency, which results in a 50% improvement over the 6T cell-based cache.

8. Conclusions

In this paper, we demonstrated how the minimum supply voltage (Vccmin) is a critical parameter that affects microprocessor power and reliability. We proposed two novel architectural techniques that enable microprocessor caches (L1 and L2) to operate despite very high memory cell failures rates, thereby allowing a processor to decrease its Vccmin to 500mV while having minimal impact on the high-performance mode. The Word-disable scheme combines two consecutive cache lines, in low-voltage mode, to form a single cache line where only non-failing words are used. The Bit-fix scheme uses a quarter of the ways in a cache set, in low-voltage mode, to store positions and fix bits for failing bits in other ways of the set. We show that the Word-disable and Bit-fix schemes enable reliable operation below 500mV in exchange for a 50% and 25% loss of cache capacity, respectively. Compared to an unrealistic baseline in which all cache lines are reliable in the low-voltage mode, our schemes reduce performance by only 10%.

Acknowledgements

We are especially grateful to Prof. Kaushik Roy and Jaydeep Kulkarni for helping us with the bit failure data and feedback. We thank Tingting Sha, Eric Ernst and the anonymous reviewers for their feedback on our work.

References

[1] A. Agarwal, et. al., "Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture," *IEEE Journal of Solid-state Circuits*, Vol. 40, no. 9, pp. 1804-1814, September, 2005.

[2] A. J. Bhavnagarwala, X. Tang, J. D. Meindl, "The Impact of intrinsic device fluctuations on CMOS SRAM Cell stability," *IEEE Journal of Solid-state Circuits*, Vol. 40, no. 9, pp. 1804-1814, September, 2005.

[3] J. Chang, et. al., "The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel® Xeon Processor 7100 Series," *IEEE Journal of Solid-state Circuits*, Vol. 42, no. 4, pp. 846-852, April, 2007.

[4] J. Doweck, "Inside the Core™ Microarchitecture," In *Proceedings of the 18th IEEE HotChips Symposium on High-Performance Chips*, August, 2006.

[5] H. Y. Hsiao, D.C. Bossen, R. T. Chien, "Orthogonal Latin Square Codes," In *IBM Journal of Research and Development,* Vol. 14, Number 4, pp. 390-394, July 1970.

[6] M. Khellah, et. al., "A 256-Kb Dual-Vcc SRAM Building Block in 65-nm CMOS Process With Actively Clamped Sleep Transistor," *IEEE Journal of Solid-state Circuits*, Vol. 42, no. 1, pp. 233-242, January, 2007.

[7] J. Kim, et. al., "Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding," To appear in the *40th International Symposium on Microarchitecture (Micro-40)*, December 2007.

[8] J. P. Kulkarni, K. Kim and K. Roy, "A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM," *IEEE Journal of Solid-state Circuits*, Vol. 42, no. 10, pp. 2303-2313, October, 2007.

[9] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of NBTI on SRAM Read Stability and Design for Reliability," *7th International Symposium on Quality Electronics Design,* pp. 6-11, March 2006.

[10] Y-H Lee, et. al., "Prediction of logic product failure due to thin gate oxide breakdown", *IEEE International Reliability Physics Symposium Proceedings,* pp. 18-28, March 2006.

[11] S. Mukhopadhyay, H. Mahmoodi and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Trabnsactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 12, pp. 1859-1880, December, 2005.

[12] S. Ozdemir, et. al., "Yield-Aware Cache Architectures," In *Proceedings of the 39th International Symposium on Microarchitecture*, pp. 15-25, December, 2006.

[13] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, 2003, pp. 657.

[14] S. E. Schuster, "Multiple Word/Bit Line Redundancy for Semiconductor Memories," *IEEE Journal of Solid-State Circuits,* Vol. SC-13, No. 5, pp. 698-703, October 1978.

[15] A. Seznec, P. Michaud, "A case for (partially) tagged Geometric History Length Branch Prediction," *Journal of Instruction Level Parallelism*, Vol. 8, Feb. 2006.

[16] P. P. Shirvani and E. J. McCluskey, "PADded Cache: A New Fault-Tolerance Technique for Cache Memories." In *Proceedings of the 17th IEEE VLSI Test Symposium,* April, 1999.

[17] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*, Cambridge University Press, 1998, pp. 144.