Chapter 6

Reliability and Aging Analysis on SRAMs Within Microprocessor Systems

Taizhi Liu, Chang-Chih Chen and Linda Milor

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72779

Abstract

The majority of transistors in a modern microprocessor are used to implement static random access memories (SRAM). Therefore, it is important to analyze the reliability of SRAM blocks. During the SRAM design, it is important to build in design margins to achieve an adequate lifetime. The two main wearout mechanisms that increase a transistor's threshold voltage are bias temperature instability (BTI) and hot carrier injections (HCI). BTI and HCI can degrade transistors' driving strength and further weaken circuit performance. In a microprocessor, first-level (L1) caches are frequently accessed, which make it especially vulnerable to BTI and HCI. In this chapter, the cache lifetimes due to BTI and HCI are studied for different cache configurations, namely, cache size, associativity, cache line size, and replacement algorithm. To give a case study, the failure probability (reliability) and the hit rate (performance) of the L1 cache in a LEON3 microprocessor are analyzed, while the microprocessor is running a set of benchmarks. Essential insights can be provided from our results to give better performance-reliability tradeoffs for cache designers.

Keywords: reliability analysis, SRAM stability, cache configurations, microprocessors, semiconductor microelectronics, very-large-scale integration (VLSI)

1. Introduction

As smaller technology nodes bring significant benefits like more density and lower power consumptions, they also pose significant reliability challenges. Not only do the manufacturing variations make the resulting transistors unreliable at low-voltage operation but also they take less time to wear out, making them more prone to failures in the field. The increasing reliability concerns hold for all types of microelectronic devices from electronics used in aerospace applications where reliability requirement is extremely critical, to mobile devices where product reliability can strongly affect market share.

BTI and HCI are two of the most dominating wearout mechanisms that increase the threshold voltage (V_{th}) of a transistor. As a result of BTI and HCI, the driving strengths of the aged transistors are weakened, which eventually could cause timing violations and faulty operation. During the static-stress window when a transistor is kept ON, BTI kicks in. There are two forms of BTI: Negative BTI (NBTI) and Positive BTI (PBTI). NBTI affects the threshold voltage of a PMOS transistor when its gate is applied LOW; and PBTI affects the threshold voltage of a NMOS transistor when its gate is applied HIGH. On the other hand, HCI happens when a transistor flips from being OFF to ON or vice versa. Therefore, HCI is more acute to those transistors that switch frequently.

In a modern microprocessor, static random access memories (SRAM) take the majority of the transistors, and thus the reliability of the SRAM cells is essential for circuit designers. Moreover, the first-level (L1) data cache is frequently accessed (read and written), making it very vulnerable to HCI. But at the same time, it also stores data for a significant amount of time, making it also vulnerable to BTI. Besides, cache efficiency is one of the most important characteristics for microprocessor system performance. Basically, for microprocessor designers, it is very important to understand both the performance and the reliability of the cache systems. There are many prior works [1–3] focused on cache architecture to improve cache efficiency. However, when different advanced techniques are used to achieve higher performance, it is still unknown how the reliability of the cache is changed. In this chapter, the failure probability of the L1 data cache is investigated for a LEON3 microprocessor when different design configurations are applied: associativity, cache line size, cache size, and replacement algorithm. We analyzed the impact of cache configurations on failure rates and cache efficiency so that cache designers can achieve performance-reliability tradeoff according to their design budgets (area, power, lifetime, etc.). We also study the impact of error correcting codes (ECC) on cache reliability.

BTI and HCI cause driving-strength mismatch in a traditional six-transistor (6T) SRAM cell. Because SRAM stability is extremely sensitive to transistor mismatches, BTI and HCI pose a significant problem to SRAM reliability [4–6]. In [7–9], the authors analyzed SRAM stability by assuming two ideal stress conditions, that is, static stress (0% or 100% duty cycle) and alternating stress (50% duty cycle). However, the realistic stress conditions of the SRAM cells really depend on customer usages (workload). In [10–13], the authors estimated the SRAM degradation due to BTI based on the realistic stress conditions considering the actual workload. On the other hand, the impact of the HCI effect on SRAM stability is not as studied as BTI because BTI is usually dominant due to its frequency independence. However, HCI is becoming more concerning as operating frequencies of nowadays chips are GHz-level [14, 15]. Some prior arts have investigated the HCI effect on SRAM cell stability [16, 17], and in [16], the simulation results are even compared with silicon experimental results.

Other research efforts have focused on balancing the amount of time that logic '0' and '1' values are stored in the cells with the aim to provide a BTI-optimal duty cycle distribution [18, 19], and by implementing redundancy into the cache design to combat BTI-induced wearout [20]. Gunadi et al. [19] also proposed to mitigate the HCI degradation by providing a uniform distribution of cache accesses across sets.

http://dx.doi.org/10.5772/intechopen.72779

In this chapter, we stress SRAM cells under different stress conditions and analyze the SRAM stability due to BTI and HCI. As a case study, the L-1 data cache of a state-of-art microprocessor (LEON3) is studied, and cache reliability and cache efficiency are analyzed by considering the realistic workload when the microprocessor is running a set of benchmarks.

2. Device-level wear-out mechanisms

We first model BTI and HCI at the device level and then abstract the models to the system level.

2.1. NBTI/PBTI

Negative BTI, as known as NBTI, is the degradation for PMOS transistors when negative gate-to-source voltage is applied. Positive BTI, known as PBTI, is the degradation of NMOS devices under positive gate-to-source voltage. Both NBTI and PBTI can cause an increase in the threshold voltage and the consequent decrease in drain current and transconductance of a MOSFET.

According to trapping/de-trapping theory [21], the threshold voltage shift (ΔV_{th}) due to BTI is modeled as a function of time under DC stress (t_{DC}):

$$\Delta V_{th}(DC) = \phi(T, E_F)\left(A + \mathrm{Bln}(t_{DC})\right) \tag{1}$$

where ϕ is proportional to the number of available traps and is a function of temperature, T, and the Fermi level, E_F, and A and B are constants. The temperature dependence of BTI is incorporated in ϕ with the Arrhenius relationship:

$$\phi(T, E_F) = \phi_0\, g(E_F)\, e^{-E_a/kT} \tag{2}$$

where k is a constant, T is temperature, and E_a is the activation energy. Since the frequency dependency of BTI has been considered as relatively insignificant, especially for low-frequency signals [22], it is not included in this work. However, the duty cycle, α, can affect the ΔV_{th}, and it is incorporated as an effective Fermi level, where $E_{F,eff} = \alpha E_{F,on} + (1-\alpha)\, E_{F,off}$. Here, $E_{F,on}$ and $E_{F,off}$ are the Fermi levels when the transistor is ON and OFF, respectively. The duty cycle accounts for the time under stress, t_{stress}, and the recovery time, t_{rec}, since $\alpha = t_{stress}/(t_{stress} + t_{rec})$. The function $g(\alpha)$ in Eq. (2) is a nonlinear function of α, which has $g(1) = 1$ and $g(0) = 0$ [21]. Overall,

$$\Delta V_{th} = \phi_0\, e^{-E_a/kT}\, g\left(t_{stress}/(t_{stress} + t_{rec})\right) \cdot \left(A + Bln(t_{stress} + t_{rec})\right) \tag{3}$$

where ϕ_0 is a constant. The constants were obtained from the experimental results in [23].

2.2. HCI

Hot carrier injection (HCI) is the phenomenon where electron or a "hole" gains sufficient kinetic energy to overcome a potential barrier necessary to break an interface state to be injected into

the gate oxide. HCI is one of the mechanisms that adversely affect the reliability of semiconductors of solid-state devices. More specifically, some of the device parameters such as the threshold voltage, channel mobility, drain saturation current, and transconductance can be degraded due to HCI. HCI was a major concern for NMOS transistors historically, and the HCI effect on PMOS transistors was relatively negligible. This was because holes have a smaller impact ionization rate than electrons, and the $Si-SiO_2$ barrier for holes is also higher than electrons. However, researchers have recently observed HCI effects on PMOS transistors [24].

As hot carriers are generated during switching of the transistors, the HCI effect is directly proportional to the switching frequency. In this chapter, we used the predictive HCI lifetime models for long-term performance-degradation simulations, where the ΔV_{th} degradations due to HCI during stress time are modeled as [25–27]:

$$\Delta V_{tp/tn} = A_{HCI}\left(r_{trans}\, t_{stress}\, t_{trans}\right)^{n} \tag{4}$$

where t_{stress} is the stress time, r_{trans} is the frequency-dependent transition rate, t_{trans} is the transition time, and A_{HCI} is a constant that depends on the inversion charge, the trap generation energy, the hot electron mean free path, and other process-dependent factors [28, 29].

3. SRAM stability

3.1. SRAM cell

Each SRAM cell can store one bit, and it is usually implemented using six transistors, which is well known as 6T SRAM cell. The structure of a 6T SRAM cell is shown in **Figure 1**. The core of the cell is formed by two CMOS inverters (the four labeled transistors in **Figure 1**), where the output potential of each inverter is fed as input into the other. The formed feedback loop stabilizes the inverters to their respective state. Besides the inverter loop, the remaining unlabeled two transistors in **Figure 1** are the access transistors, which are controlled by the word and bit lines, WL and BL, respectively. WL and BL are used to read and write from or to the cell. When the word line (WL) is low, the access transistors are turned OFF, and the cell is in standby mode. When reading, the word line (WL) is HIGH and the access transistors are

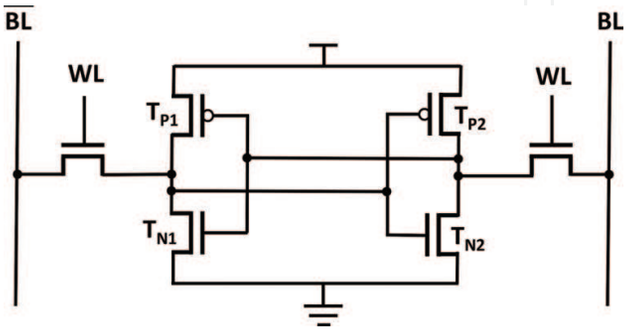

Figure 1. A typical 6T SRAM cell.

ON to allow the stored bit reflected at the bit lines. When writing, the word line (WL) is also HIGH to turn access transistors ON, and the asserted bit lines are strong enough to write the data into the inverter loop.

For the 6T SRAM cell mentioned above, all the transistors will be affected by the HCI effect during a write access when the stored bit changes. For the BTI effect, it happens when the stored bit is stable and the transistors are in static stress. More specifically, when the stored bit is a '0,' the PMOS transistor T_{P1} and the NMOS transistor T_{N2} are stressed because they are turned ON, meaning they are undergoing NBTI and PBTI, respectively. On the other hand, if a '1' is stored, the other two transistorsT_{P2} and T_{N1} are turned ON, and they are suffering from NBTI and PBTI, respectively. It is worth noting that, when one pair of transistors (T_{P1} and T_{N2}, for example) is under stress and undergoing BTI, the other pair (T_{P2} and T_{N1}) is not under stress and is under recovery from BTI degradation. However, overall, these transistors that form the inverter loop (T_{P1}, T_{N2}, T_{P2}, and T_{N1}) are continuously aging regardless of whether the cell is being read or write [30]. For the access transistors, they are only affected by BTI during the SRAM cell is being accessed (when WL is HIGH). Thus, the access transistors are much less sensitive to BTI than the inverter-loop transistors. In this chapter, we focus on the aging of the inverter-loop transistors.

3.2. Extraction of activity, temperature, IR-drop profiles

BTI and HCI effect not only depends on the time that the device is under stress but also depends on temperature. The time that the device is under stress is referred to as stress time in the following chapter. For BTI, the stress time is proportional to the duty cycle, that is, for a NMOS transistor, the stress time is equal to the total time (that the circuit is working) multiplied by the percentage so that the gate voltage is HIGH, while for PMOS transistors, it is equal to the total time multiplied by the percentage so that the gate voltage is LOW. For HCI, the stress time is proportional to the number of switching.

For the memory block within a microprocessor, it is not feasible to run SPICE simulations to get the activity (duty cycle, switching) profile of each SRAM cell. In our work, we utilize a FPGA emulation system to simulate the microprocessor. Being doing so, we are able to run benchmarks on the microprocessor and extract the activity profile in an efficient manner. Our framework to extract activity profiles is shown in **Figure 2**, which also includes the further steps to extract thermal profiles. To extract the activity profile, we synthesized the hardware RTL of the design into an FPGA and placed counters at the I/O ports of the data cache. The placed counters can track both the state probabilities (duty cycle) and the toggle rates at the I/O ports when the microprocessor is running benchmarks. The state probability is the probability of a net at each logic state, that is, logic '0' and logic '1,' and the toggle rates are the number of toggles that a net has during a unit period, for example, 1 ns. The extracted activities (state probabilities and toggle rates) were then used for activity propagation to get the complete activity profile of all the SRAM cells.

Besides activity extraction, the thermal profile throughout the microprocessor is also extracted. Moreover, because the SRAM stability strongly depends on voltage, we also consider the

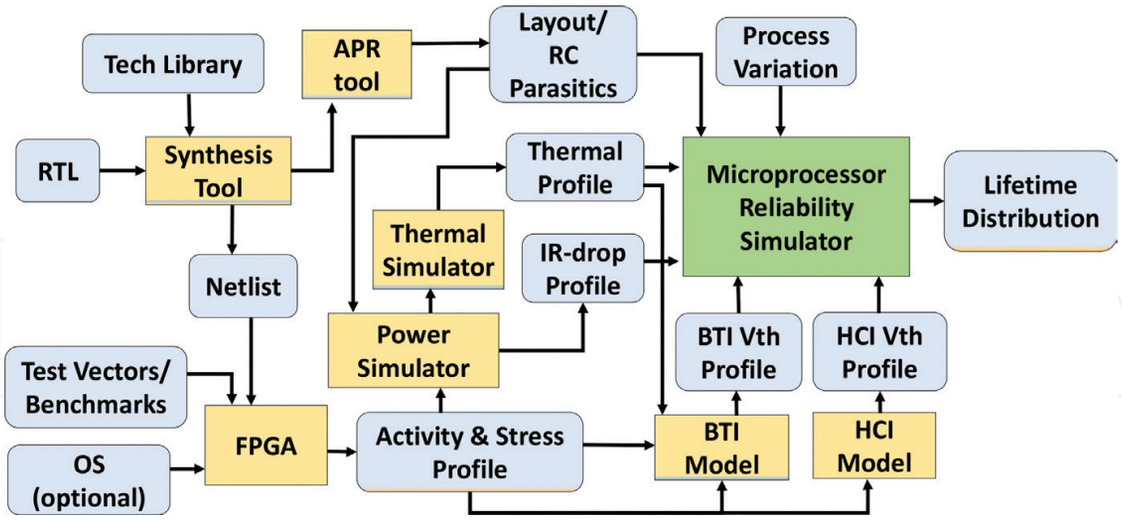

Figure 2. The FPGA-based aging assessment framework, which is used to extract the duty cycle/toggle-rate profiles, temperature profile, and the IR-drop profile.

impact of IR-drop in our work. As shown in **Figure 2**, the netlist was used for layout generation, and then RC parasitics from the layout, along with the activity profile, are fed to extract the power profile and the consequent thermal profile, using the power simulator [31] and the thermal simulator [32], respectively, for every module block of the microprocessor system.

In this chapter, we used the open-source microprocessor called LEON3 [33] as a case study. LEON3 is well known for space applications with high-level reliability requirement. We have implemented LEON3 with superscalar abilities on a commercial 90 nm technology process. The logic part of the LEON3 core includes a 32-bit multiplier (MUL), a 32-bit divider (DIV), a 32-bit general purpose integer unit (IU), and a memory management unit (MMU). The memory part of the LEON3 core consists of data caches (D-Caches) and instruction caches (I-Caches), cache tag units (Dtags and Itags), and window-based register file (RF). In this chapter, we focus our analysis on L1 D-Caches due to its importance to microprocessor performance and its high sensitivity to aging effects. The proposed method is applicable to other memory blocks as well.

Standard benchmarks from MiBench [34] were used as the microprocessor applications. **Figures 3** and **4** show the distributions of the state probabilities and the transition rates, respectively, of the data cache, when the microprocessor is running a standard benchmark. **Figure 5** shows the average temperature distribution and average IR-drop distribution when the microprocessor is running a standard benchmark.

3.3. SRAM stability degradation analysis under BTI and HCI

In this chapter, several performance metrics were used to characterize SRAM stability, including the read and retention static noise margins (SNMs), the read current (I_{READ}), the minimum retention voltage (Vdd-min-ret), and the write margin. SNM is a key figure of merit for an SRAM cell. It is the minimum DC noise voltage necessary to change the state of an SRAM cell and can be extracted by nesting the largest possible square in the two voltage transfer curves

http://dx.doi.org/10.5772/intechopen.72779

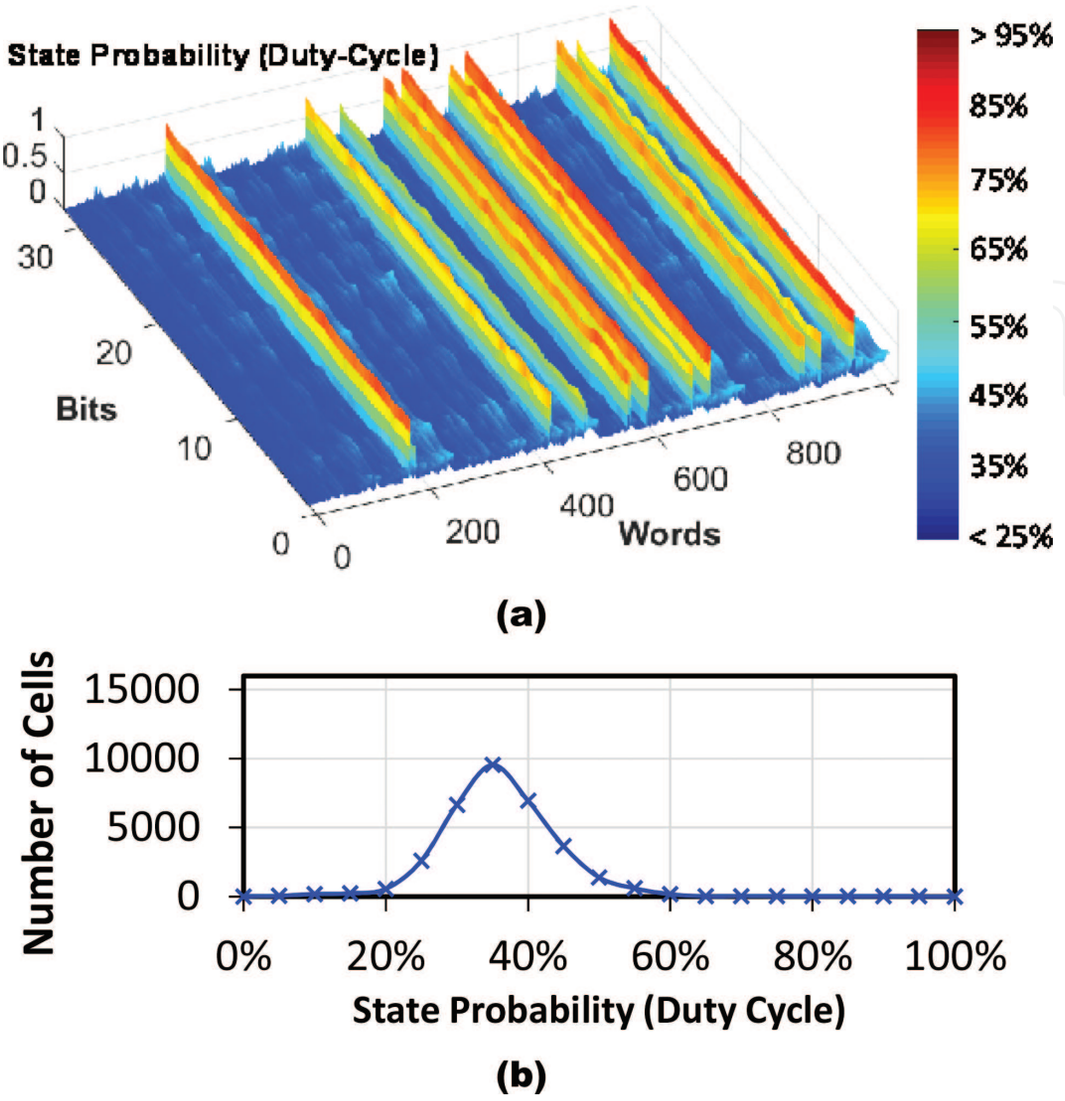

Figure 3. (a) The distribution of state probability for the 32 KB data cache shown in 1024 words and (b) the histogram of the state probability distribution in the number of SRAM cells.

(VTC) of the involved CMOS inverters [35]. The read SNM is measured when the access transistors are turned ON, while for the retention SNM, the access transistors are OFF. I_{READ} is the current flowing through pull-down transistors during a read access, and it is inversely proportional to access time. Vdd-min-ret is the minimum supply voltage that an SRAM can retain the stored bit. The write margin is the minimum voltage needed to flip the state of the cell, with the access transistors are ON. The lifetime calculations in this chapter are based on the following assumption: when any of these four metrics mentioned above has degraded to a predefined threshold, the SRAM cell is said to have failed and thus the lifetime of the cell is calculated.

In this chapter, the process variations of two important parameters, channel length and threshold voltage, are included, assuming they follow normal distribution with standard deviation equal to 10% of their corresponding nominal values.

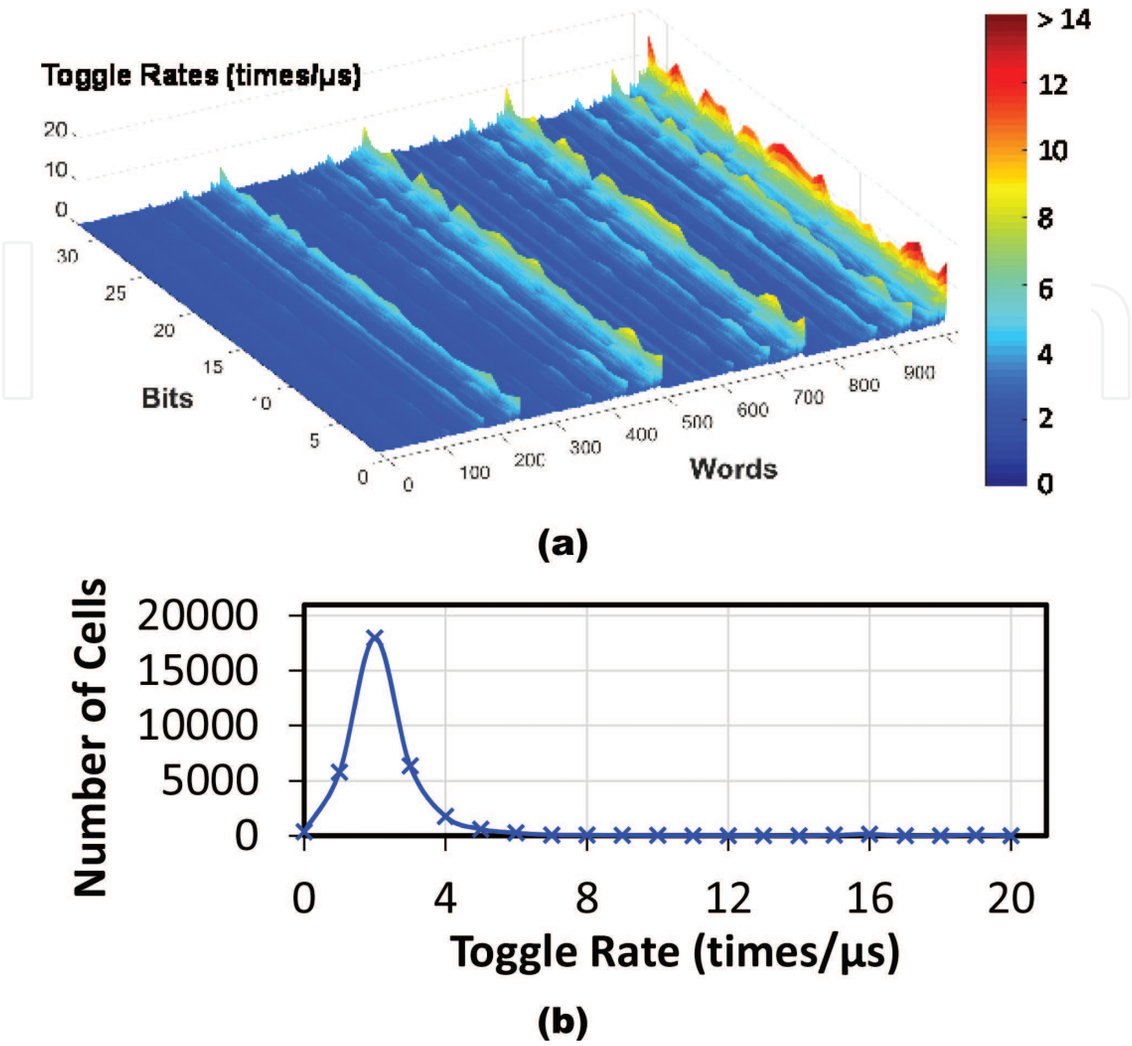

Figure 4. (a) The distribution of transition rate for the 32 KB data cache shown in 1024 words and (b) the histogram of the transition-rate distribution in the number of SRAM cells.

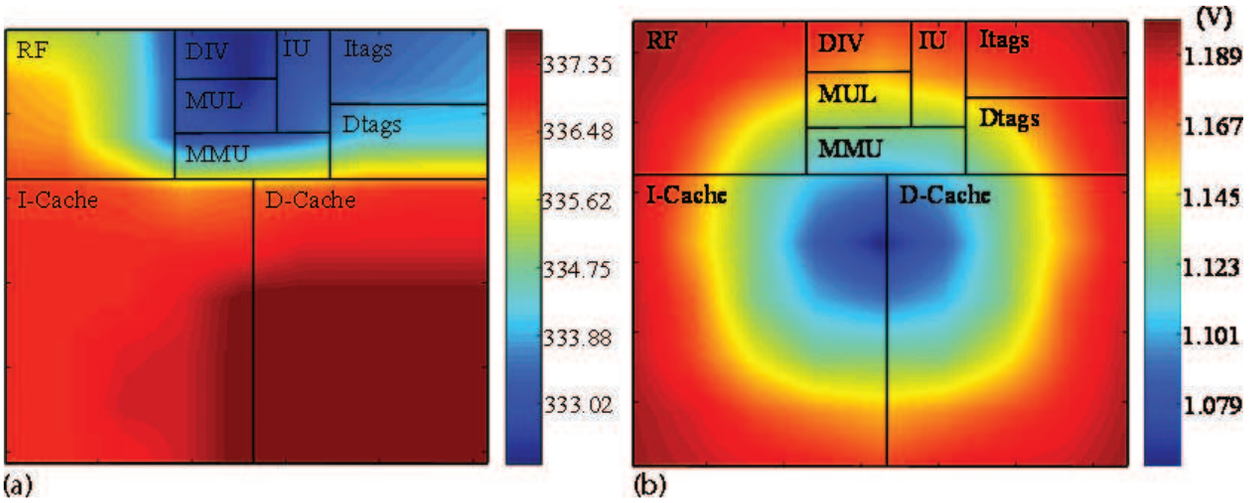

Figure 5. (a) The average temperature distribution and (b) the average IR-drop distribution of the microprocessor while running a standard benchmark.

http://dx.doi.org/10.5772/intechopen.72779

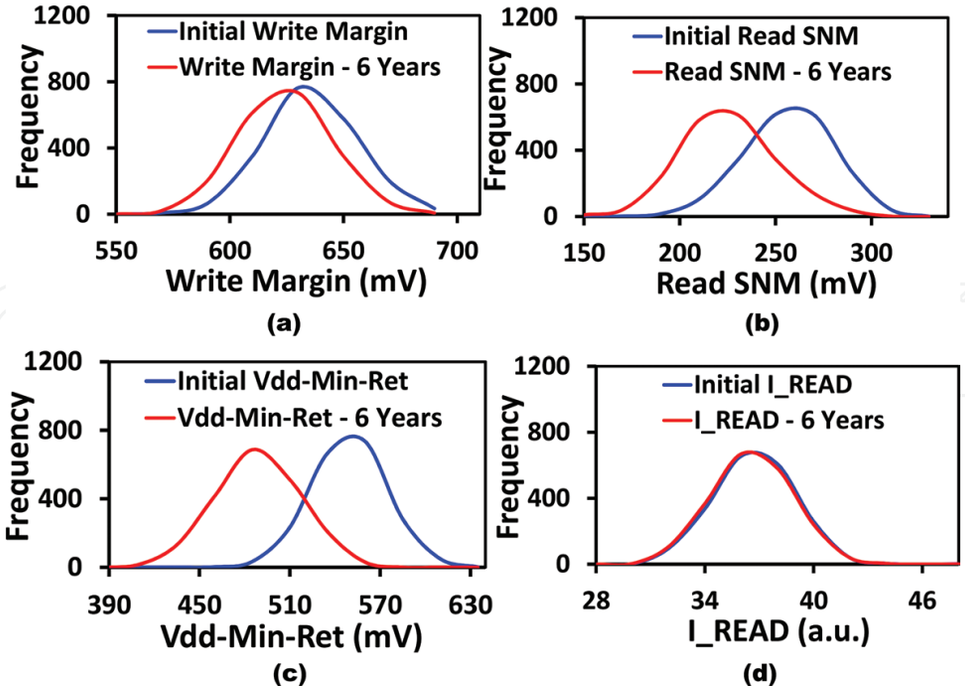

Figure 6. The degradation of the write margin, the read SNM, the Vdd-min-ret, and the I_{READ} of a memory cell due to BTI shown in (a)–(d), respectively.

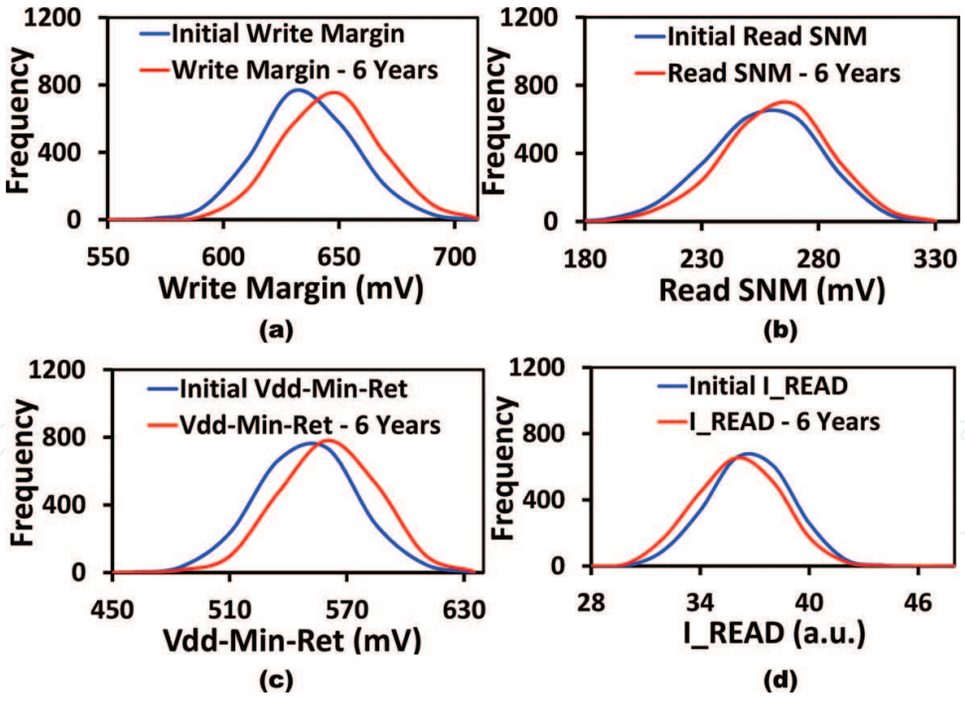

Figure 7. The degradation of the write margin, the read SNM, the Vdd-min-ret, and the I_{READ} of a memory cell due to HCI shown in (a)–(d), respectively.

Figures 6 and **7** show the degradation of the read SNM, the write margin, the Vdd-min-ret, and the I_{READ} of a memory cell due to BTI and HCI, respectively. As it is seen from **Figure 6**, BTI severely degrades the read SNM as well as the write margin. The Vdd-min-ret is also affected,

while the I_{READ} is relatively unaffected. On the one hand, HCI, as shown in **Figure 7**, only degrades I_{READ} and improves the other three cell performances. This is because the cell becomes increasingly skewed under BTI as some devices degrade more than the others. This leads to impaired noise immunity. On the other hand, all the devices undergo the same stress due to HCI, as explained in Section 3.1.

4. Lifetime analysis

4.1. Memory cell lifetime characterization

To estimate the SRAM lifetimes due to BTI and HCI, the activity profile, thermal profile, and IR-drop profile of the memory were collected by the framework as shown in Section 3.2. The stress and thermal profiles are fed into the BTI and HCI models described in Section 2 to obtain the threshold voltage degradation. Then, the thermal profile, IR-drop profile, the BTI and HCI threshold voltage degradations, together with process parameter variations, were used to analyze the degradation of SRAM stability via Monte Carlo SPICE simulations (2000 samples for each Monte Carlo run). As mentioned in Section 3.3, an SRAM cell is assumed to have failed when any of the aforementioned four metrics degrades the predefined threshold levels. Then, the lifetime of the SRAM cell is obtained by interpolating the two time stamps where the failure happens in between. To characterize the cell lifetime, the cell is simulated 2000 times for each of the time stamps in SPICE. The time stamps basically define the level of BTI/HCI degradations, that is, the BTI/HCI-induced threshold voltage shifts are back annotated to the SPICE netlist for Monte Carlo simulations.

If we run Monte Carlo SPICE simulations for each cell for each time stamp, it would be very time-consuming and not practical. To address the large number of cells, the state probabilities and toggle rates are partitioned into 21 stress states (0%, 5%, 10%, …, 95%, 100%) for BTI and HCI, respectively. This strategy can dramatically reduce the cost of SPICE simulation time while not giving up too much accuracy. It is straightforward to assume that cells from the same stress state share the same state probability and the same toggle rate. Furthermore, all the cells in one stress state share the same lifetime distribution.

For BTI, the stress states are partitioned by state probabilities. The 0% stress state means that 0% of time the cell is storing a '1,' while the 100% stress state means a '1' is stored all the time. For HCI, the stress states are the percentage of the maximum toggle rate that we observed, that is, 0%, 5%, 10%, …, 100% of the maximum toggle rate. **Figures 3(b)** and **4(b)** show an example for the stress-state distribution for BTI and HCI, respectively, for a 32 KB data cache. The stress-state distribution not only depends on the benchmark that is running but also depends on the configuration of the cache system. We will discuss this impact in Section 5.

As process variations are considered, the lifetime of each SRAM cell is now a distribution rather than a value. With Monte Carlo simulations, the lifetime distribution is computed for each stress state. Importance sampling [36] was employed to have sufficient samples for the tail part of the distribution. **Figures 8** and **9** show the lifetime distributions for five representative stress

http://dx.doi.org/10.5772/intechopen.72779

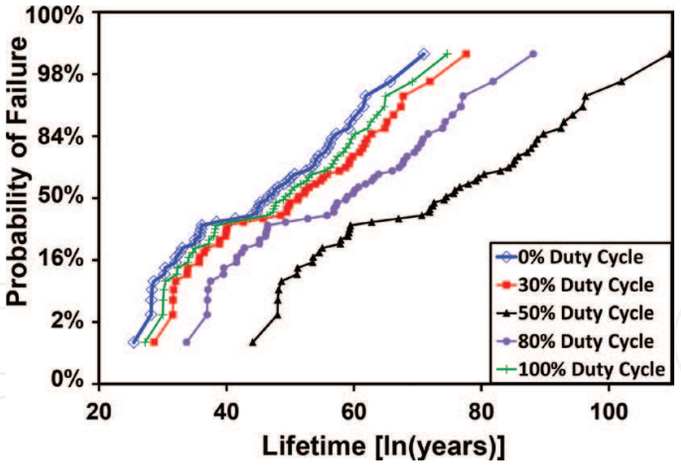

Figure 8. The BTI lifetime distribution of an SRAM cell when it is under a specific duty cycle stress state. Five duty cycle stress states are shown as follows: 0%, 30%, 50%, 80%, and 100%.

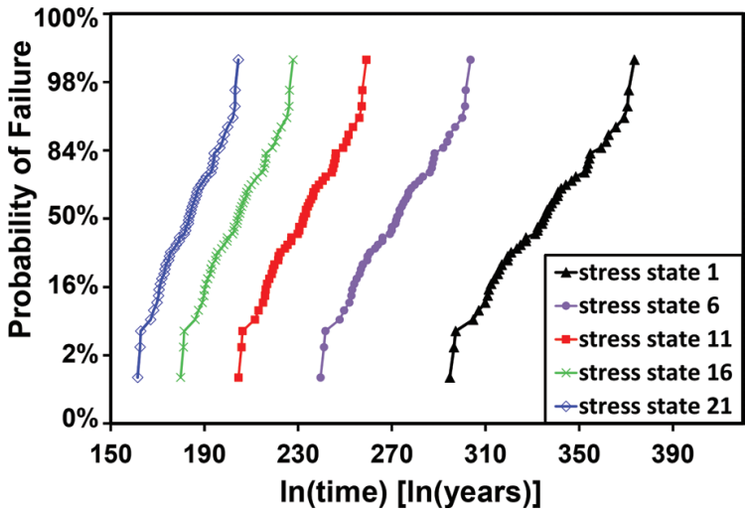

Figure 9. The HCI lifetime distribution of an SRAM cell when it is under a specific toggle-rate stress state. Five toggle-rate stress states are shown: State N means a toggle rate of N times/μs.

states, for BTI and HCI, respectively. As shown, for BTI, 50% stress state has the best lifetime, while for HCI, the lowest switching rate results in the best lifetime.

Log-normal distribution is the best fit for the lifetimes in **Figures 8** and **9**. Once the fitted log-normal distributions are determined, it is straightforward to obtain the failure rate of an SRAM cell, PF_{bit}, as a function of time, t:

$$PF_{bit} = Probability\ of\ (Lifetime < t). \tag{5}$$

Then, the failure probability of a word can be calculated, assuming no error correction codes:

$$PF_{word} = 1 - \prod_{i=1}^{N} \left(1 - PF_{bit_i}\right) \tag{6}$$

where N is the number of bits in one word, PF_{word} is the failure probability of a word, and PF_{bit} is the failure probability of a bit. Without ECC, we can safely assume that if there is one cell fails to work, the whole memory system will fail. It is then straightforward to get the failure probability of the whole SRAM block:

$$PF_{SRAM} = 1 - \prod_{i=1}^{N_{word}} \left(1 - FP_{word_i}\right) \tag{7}$$

where N_{word} is the number of words, PF_{SRAM} is the failure probability of of the memory block, FP_{word_i} is the probability of failure of i -th word. As PF_{bit} is a function of time, PF_{word} is also a function of time, and so is PF_{SRAM}.

The inclusion of error correcting codes can detect and correct the internal data corruption in SRAMs. In this chapter, BCH codes [37] were used, which consumes seven additional bits per word and can correct one bit per word. With ECC, for a word containing N bits (including ECC), the failure probability of a word, F_{word}, is different from Eq. (6):

$$PF_{word} = 1 - \prod_{i=1}^{N} \left(1 - PF_{bit_i}\right) - \sum_{j=1}^{N} \left[PF_{bit_j} * \prod_{i \neq j} \left(1 - PF_{bit_i}\right)\right] \tag{8}$$

In LEON3, the word size is $N = 32$ for the data cache without error correcting codes (ECC). With ECC, the word size is $N = 39$. Note that Eqs. (5) and (7) are the same for with ECC and without ECC.

5. Performance-reliability analysis for different cache configurations

In this section, we study the impact of cache configurations on cache reliability. Four categories are considered, including cache associativity, cache size, cache line size, and the replacement algorithm. The cache hit rates are also presented along with the cache reliability to analyze the performance-reliability tradeoffs. Besides, we also show the impact of error correction codes (ECC) on cache reliability.

Six benchmarks from MiBench [34] are tested: Qsort, SHA, CRC32, FFT, Basicmath, and Dijkstra. Qsort benchmark implements the classical Qsort algorithm on a large array of strings. SHA benchmark produces a 160-bit digest for a given input by using the classical secure hash algorithm. CRC32 benchmark performs a 32-bit Cyclic Redundancy Check (CRC) to detect errors in data transmission. FFT benchmark performs a fast Fourier transform on an array of data. Basicmath benchmark has many basic mathematical calculations, which usually do not have dedicated hardware support in embedded processors. Finally, the Dijkstra benchmark implements the well-known Dijkstra's algorithm to get the shortest path between every pair of nodes on a large graph, which is stored in an adjacency matrix.

The state-probability (duty cycle) distributions are shown in **Figure 10**, for each of the six benchmarks mentioned above. It can be obviously seen that the distributions are leaning to the left. It is because in data cache memory, logic '0' is more dominant than logic '1' [38]. In fact, memory is typically initialized to all '0's when allocated. This means, even if the benchmark is

http://dx.doi.org/10.5772/intechopen.72779

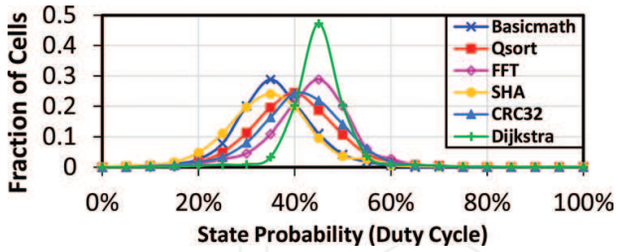

Figure 10. The duty cycle distributions of SRAM cells in a two-way 32 KB data cache while the microprocessor is running six different benchmarks.

writing a '0' and '1' to any bit with equal likelihood, logic '0' is always stored longer than logic '1'. There are some other reasons for '0' being stored longer, including false Boolean values and NULL pointers are represented with '0's, and most data in dense-form sparse matrices are '0's [39].

In our setup, the microprocessor is running at 250 MHz frequency. For this level of frequency, BTI is dominant and the HCI effect has a smaller impact. This is because that BTI is independent of frequency, while HCI is frequency dependent and 250 MHz is not a very high frequency. However, the HCI effect would be more impactful if the microprocessor is working at higher frequencies.

The overall failure probability of the SRAM block is calculated based on the following equation:

$$PF_{SRAM.Overall} = 1 - \left(1 - PF_{SRAM.BTI}\right) * \left(1 - PF_{SRAM.HCI}\right). \quad (9)$$

where $PF_{SRAM.BTI}$ is the failure probability due to BTI, and $PF_{SRAM.HCI}$ is the failure probability due to HCI.

5.1. Associativity

There are three types of cache associativity: fully associative, direct mapped, and n-way set associative. For fully associative, data could be anywhere in the cache, making it very expensive to implement as it must check the tag of every cache line. For direct mapped, data can only go to a single cache line in the cache based on the memory address of the data. Set associative cache is a trade-off between direct mapped cache and fully associative cache. The cache is divided into 'n' sets, and each set contains a number of cache lines. Four-way set associative means the cache is divided into sets that can fit four blocks each, while a two-way set associative means each set can hold two blocks. From this perspective, a fully associative cache of m cache lines is m-way set associative, and a direct mapped cache is actually 1-way set associative. Although higher associativity can achieve higher hit rate, it is more expensive in terms of timing and area cost.

In our work, we have implemented the LEON3 data cache with three different associativities: 1-way, 2-way, and 4-way. Other configurations are kept the same: 16-byte cache line size, 32 KB cache size, and LRU replacement algorithm.

Figure 11 shows the failure probability of the whole data cache for two illustrative benchmarks: Basicmath and Dijkstra (other benchmarks have a similar trend). The hit rates for

1-way, 2-way, and 4-way associativity for Basicmath are 96.12%, 96.33%, 96.36%, respectively. For Dijkstra, they are 62.23%, 64.81%, and 65.54%, respectively. It is seen from the results that although higher associativity can get higher hit rates, it adversely impacts the reliability.

5.2. Cache line size

When the processor accesses a part of memory that is not already in the cache, it loads a chunk of the memory around the accessed address into the cache, hoping that it will soon be used again. When data are transferred between cache and main memory, this chunk of data is handled in a fixed size, called cache lines. A cache can only hold a limited number of lines, determined by the cache size. For example, a 64 KB cache with 64-byte lines has 1024 cache lines. In LEON3, cache line size can be configured as 16-byte or 32-byte. Other configurations are kept the same: two-way set associative, 32 KB cache size, and LRU replacement algorithm.

Figure 12 shows the failure probabilities for 16-byte and 32-byte cache line size for the six tested benchmarks. It is obviously seen that, for all the tested benchmarks, 32-byte cache line has lower failure probability than 16-byte, meaning 32-byte configuration is more reliable than 16-byte. Besides, 32-byte also achieves better hit rates than 16-byte for four of the six

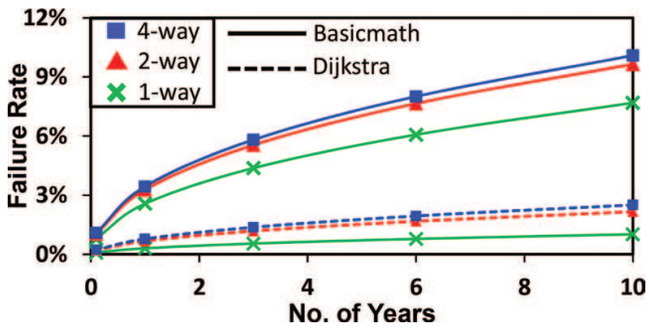

Figure 11. The failure probability as a function of time for three different associativities and two benchmarks.

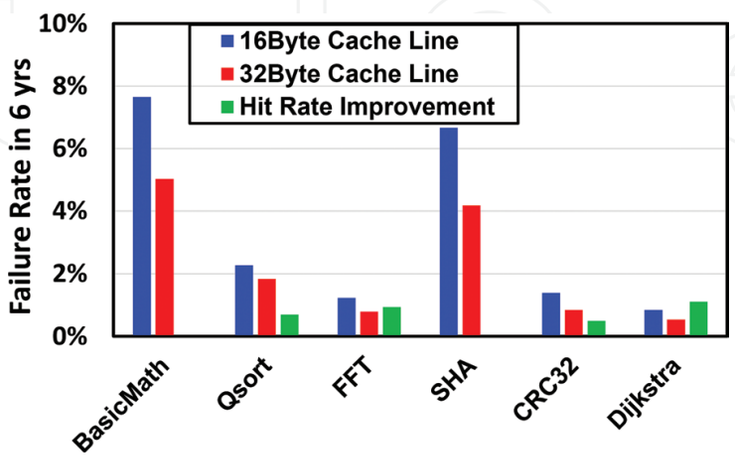

Figure 12. The failure probabilities in 6 years for 16-byte cache line and 32-byte cache line for six applications. The hit-rate improvement is also shown, defined as the improvement of using 32-byte cache line compared to 16-byte line.

benchmarks except for SHA and Basicmath, and hit rates for 32-byte and 16-byte are almost the same. Overall, from our observation, larger cache line size can improve both hit rate and reliability.

The reason for that is, a cache miss in a 32 Byte cache line can produce more recovery cycles up to 256 (32 × 8) SRAM cells, which is twice as with a 16-byte cache line (16 × 8 SRAM cells). The more BTI recovery cycles, the better reliability the cache would have.

5.3. Cache size

In our experiments, we have set five different cache sizes for the data cache of LEON3: 4, 16, 32, 64, and 128 KB. Other configurations are kept the same: two-way set associative, 16-byte cache linesize, and LRU replacement algorithm. In **Figure 13**, the hit rate and probability that the data cache fails in 6 years are presented for different cache sizes. As expected, the larger the cache size, the cache is more vulnerable and less reliable. For hit rate, although larger cache size always results in better hit rates, the improvement is little when cache size is larger than 32 KB. It is also worth noting that larger cache size causes more area and more power.

5.4. Replacement algorithm

If all the cache lines in the cache are in use, when the microprocessor accesses a new line, one of the lines currently in the cache must be evicted to make room for the new line. The policy that the microprocessor uses to choose the entry to evict is called the replacement policy.

The heuristic of any replacement policy is that it tries to predict which existing entry is the least likely to be used in the future. The most common replacement policy in modern processors is least recently used (LRU) policy. The Least-Recent-Replaced (LRR) algorithm evicts the cache entry, which is least recently replaced. Another replacement policy is random replacement, meaning that a random cache line is selected for eviction. Among them, random

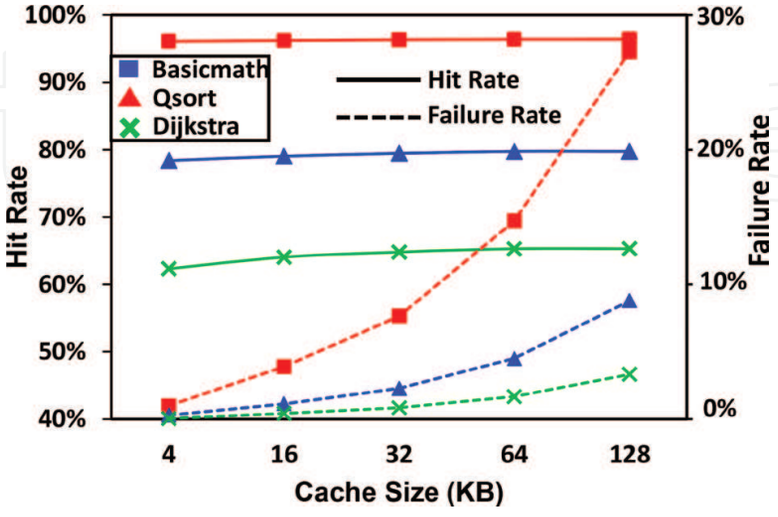

Figure 13. The hit rate and the failure probability in 6 years are shown for five different cache sizes and for three applications.

replacement policy is the simplest. It has low area overhead but suffers from poor cache efficiency. LRR algorithm uses one extra bit in the tag part, and it also has low area overhead. LRU algorithm typically has the best performance but with the cost of the highest area overhead among the three.

In this chapter, we have configured LEON3 to three different replacement algorithms, LRR, LRU, and Random. Other configurations are kept the same: two-way set associative, 16-byte cache line size, and 32 KB cache size.

Figure 14 shows the failure probabilities for the three replacement algorithms as well as the hit-rate improvement of LRU and LRR compared to Random. As expected, LRU has the best hit rate for all the tested benchmarks. However, seen from the results, it has lower reliability compared to LRR and Random. The reason for the abovementioned results is LRU has better hit rate and fewer misses, which result in fewer recovery cycles.

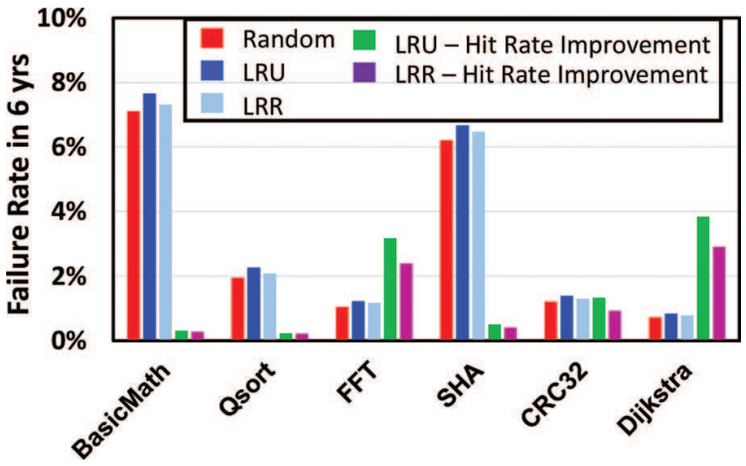

Figure 14. The failure probabilities in 6 years for 16-byte cache line and 32-byte cache line for six applications. The hit-rate improvement is also shown, defined as the improvement of using 32-byte cache line compared to 16-byte line.

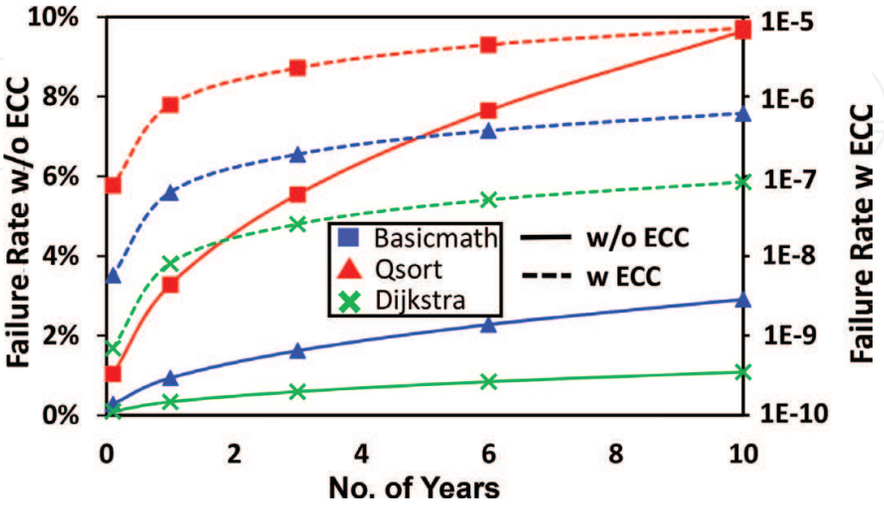

Figure 15. The failure probabilities of the two-way 32 KB data cache with and without ECC codes are shown as a function of time for three applications.

5.5. Error correcting codes

Error correcting codes (ECC) is used to detect and correct internal data corruptions in SRAMs. It uses some extra bits to check the data consistency and to correct the corrupted data. As mentioned, BCH codes [37] was used which consumes seven additional bits per word and can correct one bit per word, meaning the number of bits per word is 39 with the inclusion of ECC for LEON3.

Figure 15 shows the failure probabilities of the data cache for with and without ECC. Again, the failure probabilities are a function of time. Three illustrative benchmarks are present (other benchmarks have similar results). As shown in the results, ECC can significantly improve cache reliability.

6. Insights and conclusions

We have shown the reliability and performance of the data cache for different configurations. For associativity, larger associativity has better performance but worse reliability. According to the results, two-way set associative cache achieves the optimal performance-reliability balance. For cache line size, 32-byte cache line is better than 16-byte in both performance and reliability. Cache size is of great significance to cache reliability. We also observed that when cache size increases larger than 16 KB, the cache reliability dramatically drops while the performance (hit rate) has very limited improvement. For replacement algorithm, 'Random' replacement policy has the worst hit rate but the best reliability, while the popular LRU algorithm has the best hit rate but the worst reliability among the three. Therefore, tradeoffs can be made between the three replacement algorithms. ECC always improves reliability with area and power overhead.

Overall, experimental results show that the cache size and ECC codes are of great significance to cache reliability, while other metrics have smaller impact. According to the performance-reliability evaluation, an optimal tradeoff could be achieved for the cache design in a microprocessor system.

Author details

Taizhi Liu*, Chang-Chih Chen and Linda Milor

*Address all correspondence to: taizhiliu88@gatech.edu

Georgia Institute of Technology, Georgia

References

[1] Jouppi NP et al. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In: Proceedings of 17th International Symposium on Computer Architecture (ISCA-17). 1990. pp. 364-373

[2] Albonesi DH. Selective cache ways: On-demand cache resource allocation. In: Proceedings of 32nd International Symposium on Microarchiteure (MICRO-32). 1999. pp. 248-259

[3] Jaleel A et al. High performance cache replacement using re-reference interval prediction (RRIP). In: Proceedings of 37th International Symposium on Computer Architecture (ISCA-37). 2010. pp. 60-71

[4] Kaczer B et al. Atomistic approach to variability of bias-temperature instability in circuit simulations. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2011. pp. XT.3.1-XT.3.5

[5] Huard V et al. NBTI degradation: From transistor to SRAM arrays. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2008. pp. 289-300

[6] Bansal A et al. Impact of NBTI and PBTI in SRAM bit-cells: Relative sensitivities and guidelines for application-specific target stability/performance. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2009. pp. 745-749

[7] Lin JC et al. Time dependent Vccmin degradation of SRAM fabricated with high-k gate Dielectris. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2007. pp. 439-444

[8] Kang K et al. Impact of negative-bias temperature instability in nanoscale SRAM array: Modeling and analysis. In: TCAD. 2007. pp. 1770-1781

[9] Bansal A et al. Impacts of NBTI and PBTI on SRAM static/dynamic noise margins and cell failure probability. Journal Microelectronics and reliability. 2009;**49**:642-649

[10] Bansal A, Kim J-J, Rao R. Usage-based degradation of SRAM arrays due to bias temperature instability. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2012. pp. 2F.6.1-2F.6.4

[11] Weckx P et al. Defect-based methodology for workload-dependent circuit lifetime projections-Application to SRAM. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2013. pp. 3A.4.1-3A.4.7

[12] Angot D et al. The impact of high Vth drifts tail and real workloads on SRAM reliability. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2014. pp. CA.10.1-CA.10.6

[13] Mintarno E et al. Workload dependent NBTI and PBTI analysis for a sub-45 nm commercial microprocessor. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2013. pp. 3A.1.1-3A.1.6

[14] Khan S et al. Trends and challenges of SRAM reliability in the nano-scale era. In: Proceedings of Design and Technology of Integrated Systems in Nanoscale Era (DTIS). 2010. pp. 1-6

[15] Indaco M et al. On the impact of process variability and aging on the reliability of emerging memories (embedded tutorial). In: Proceedings of European Test Symposium (ETS). 2014. pp. 1-10

[16] Huard V et al. Managing SRAM reliability from bitcell to library level. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS). 2010. pp. 655-664

[17] Qin J et al. SRAM stability analysis considering gate oxide SBD, NBTI and HCI. In: Proceedings of International Integrated Reliability Workshop (IIRW). 2007. pp. 33-37

[18] Siddiqua T et al. Recovery boosting: A technique to enhance NBTI recovery in SRAM arrays. In: Proceedings of IEEE Computer Society Annual Symposium VLSI. 2010. pp. 393-398

[19] Gunadi E et al. Combating aging with the colt duty cycle equalizer. In: Proceedings of 43rd International Symposium on Microarchiteure (MICRO-43). 2010. pp. 103-114

[20] Shin J et al. A proactive Wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime. In: Proceedings of 35th International Symposium on Computer Architecture (ISCA-35). 2008. pp. 353-362

[21] Wirth GI, da Silva R, Kaczer B. Statistical model for MOSFET bias temperature instability component due to charge trapping. IEEE Transactions on Electron Devices. 2011;**58**(8):2743-2751

[22] Fernandez R, Kaczer B, Nackaerts A, Demuynck S, Rodriguez R, Nafria M, Groeseneken G. AC NBTI studies in the 1 Hz–2 GHz range on dedicated on-chip CMOS circuit. In: Proceedings of International Electron Devices Meeting. 2006

[23] Zafar S, Kim YH, Narayanan V, Cabral C, Paruchuri V, Doris B, Stathis J, Callegari A, Chudzik M. A comparative study of NBTI and PBTI (charge trapping) in S_iO_2/HFO_2 stacks with FUSI, TiN, Re Gates. In: Proceedings of Symposium VLSI Technology. 2006. pp. 23-25

[24] Chen S-Y, Tu C-H, Kao P-W, Lin M-H, Haung H-S, Lin J-C, Wang M-C, Wu S-H, Jhou Z-W, Chou S, Ko J. Investigation of DC hot-carrier degradation at elevated temperatures for p-channel metal-oxide-semiconductor field-effect transistors. Japanese Journal of Applied Physics. 2008;**47**(3):1527-1531

[25] Wang W, Reddy V, Krishnan AT, Vattikonda R, Krishnan S, Cao Y. Compact modeling and simulation of circuit reliability for 65-nm CMOS technology. IEEE Transaction on Device and Materials Reliability. 2007;**7**(4):509-517

[26] Liu T, Chen C-C, Cha S, Milor L. System-level variation-aware aging simulator using a unified novel gate-delay model for bias temperature instability, hot carrier injection, and gate oxide breakdown. Microelectronics Reliability. 2015. DOI: 10.1016/j.microrel.2015.06.008.

[27] Liu T, Chen C-C, Kim W, Milor L. Comprehensive reliability and aging analysis on SRAMs within microprocessor systems. Microelectronics Reliability. 2015. DOI: 10.1016/j.microrel.2015.06.078

[28] Ma C, Li B, Zhang L, He J, Zhang X, Lin X, Chan M. A unified FinFET reliability model including high K gate stack dynamic threshold voltage, hot carrier injection, and negative bias temperature instability. In: Proceedings of International Symposium Quality Electronic Design (ISQED). 2009. pp. 7-12

[29] Tu CH, Chen SY, Chuang AE, Huang HS, Jhou ZW, Chang CJ, Chou S, Ko J. Transistor variability after CHC and NBTI stress in 90 nm pMOSFET technology. Electronics Letters. 2009;**45**(15):854-856

[30] Calimera A et al. Partitioned cache architectures for reduced NBTI-induced aging. In: Proceedings of DATE. 2011. pp. 1-6

[31] PrimeTime Power Modeling Tool. [Online]. Available: http://www.synopsys.com/Tools/Implementation/SignOff/PrimeTime/Pages/default.aspx [Accessed June, 2014]

[32] HotSpot Temperature Modeling Tool. [Online]. Available: http://lava.cs.virginia.edu/HotSpot [Accessed March, 2014]

[33] LEON3 Processor. Available: http://gaisler.com/index.php/downloads/leongrlib [Accessed December, 2015]

[34] Mibench benchmark: http://www.eecs.umich.edu/mibench

[35] Seevinck E, List FJ, Lohstroh J. Static-noise margin analysis of MOS SRAM cells. IEEE Journal of Solid-State Circuits. 1987;**22**(5):748-754

[36] Kang R, Joshi R, Nassif S. Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events. In: Proceedings of Design Automation Conference. 2006. pp. 69-72

[37] Sklar B, Harris FJ. The ABCs of linear block codes. IEEE Signal Processing Magazine. 2004;**21**:14-35

[38] Ricketts A et al. Investigating the impact of NBTI on different power saving cache strategies. In: Proceedings of DATE. 2010. pp. 592-597

[39] Pekhimenko G et al. Base-delta-immediate compression: practical data compression for on-chip caches. In: Proceedings of of 21st International Conference on Parallel Architecture and Compilation Techniques (PACT-12). 2012. pp. 377-388