On Microarchitectural Mechanisms for Cache Wearout Reduction

Alejandro Valero, Negar Miralaei, Salvador Petit, *Member, IEEE,* Julio Sahuquillo, *Member, IEEE,* and Timothy M. Jones, *Member, IEEE*

***Abstract*—Hot Carrier Injection (HCI) and Bias Temperature Instability (BTI) are two of the main deleterious effects that increase a transistor's threshold voltage over the lifetime of a microprocessor. This voltage degradation causes slower transistor switching and eventually can result in faulty operation. HCI manifests itself when transistors switch from logic '0' to '1' and vice versa, whereas BTI is the result of a transistor maintaining the same logic value for an extended period of time. These failure mechanisms are especially acute in those transistors used to implement the SRAM cells of first-level (L1) caches, which are frequently accessed, so they are critical for performance, and they are continuously aging. This paper focuses on microarchitectural solutions to reduce transistor aging effects induced by both HCI and BTI in the data array of L1 data caches. First, we show that the majority of cell flips are concentrated in a small number of specific bits within each data word. In addition, we also build upon previous studies showing that logic '0' is the most frequently written value in a cache by identifying which cells hold a given logic value for a significant amount of time. Based on these observations, this work introduces a number of architectural techniques that spread the number of flips evenly across memory cells and reduce the amount of time that logic '0' values are stored in the cells by switching off specific data bytes. Experimental results show that the threshold voltage degradation savings range from 21.8% to 44.3% depending on the application.**

***Index Terms*—BTI, cache memories, cell flips, duty cycle distribution, HCI, threshold voltage degradation.**

I. Introduction

MODERN day computer systems have benefited from being designed and manufactured using an ever-increasing budget of transistors on very reliable integrated circuits. However, as technology moves forward, such a "free lunch" is over as increasingly smaller technology nodes pose significant reliability challenges. Not only do variations in the manufacturing process make the resulting transistors unreliable at low voltage operation, but they take less and less time to wear out, decreasing their lifetimes (from tens of years in current systems to 1-2 years or fewer in the near future [1]) and making them more prone to failures in the field. Thus, lifetime reliability must be treated as a major design constraint. This concern holds for all kinds of computing devices, ranging from server processors to embedded systems like tablets and mobiles, where lifetime is an assertive requirement and the market share strongly depends on their reliability.

The two main phenomena that speed up aging are referred to as Hot Carrier Injection (HCI) and Bias Temperature Instability (BTI). The former effect increases with transistor activity over the lifetime of the processor; that is, when a transistor flips from being on to off and vice versa, leading to threshold voltage (V_{th}) degradation, which in turn causes an increase in transistor switching delay and can result in timing violations and faulty operation when the critical paths become longer than the processor's clock period. Overall, HCI is accentuated in the microprocessor components with frequent switching. On the other hand, BTI accelerates transistor degradation when a transistor is kept on for a long time, and takes two forms: Negative BTI (NBTI), which affects PMOS transistors when a '0' is applied to the gate; and Positive BTI (PBTI), which affects NMOS when a '1' is applied.

A significant amount of the transistors in most modern chip multiprocessors are used to implement SRAM storage along the cache hierarchy [2]. Therefore, it is important to target these structures to slow down aging. The first-level (L1) data cache is a prime candidate since it is regularly written, yet stores data for significant amounts of time. Besides, its availability is critical for system performance. The SRAM cell transistors are stressed by HCI and BTI when the stored logic value flips and when it is retained for a long period without flipping (i.e., a duty cycle), respectively. Note that these situations are strongly related to each other. Thus, a given technique designed to exclusively attack BTI might exacerbate HCI as a side effect, and vice versa.

Prior architectural research has analyzed cache degradation mainly due to BTI effects. There have been some attempts to diminish BTI aging by periodically inverting the stored logic values in the cells [3], [4], [5], by implementing redundant cell regions in the cache [6], and by reducing the cache supply voltage [7]. In [8], authors propose a tentative approach to combat BTI and HCI by balancing the cache utilization. However, the cache contents are flushed from time to time, which might incur in significant performance degradation.

Unlike previous works, we extensively analyze the data patterns of the stored contents in L1 data caches in terms of how they affect BTI and HCI and, based on the results of this study, we propose microarchitectural mechanisms to extend

A. Valero is with the Department of Computer and Systems Engineering, University of Zaragoza, Calle María de Luna 1, 50018 Zaragoza, Spain, e-mail: alvabre@unizar.es.

S. Petit and J. Sahuquillo are with the Department of Computer Engineering, Universitat Politècnica de València, Camí de Vera S/N, 46022 Valencia, Spain, e-mail: {spetit, jsahuqui}@disca.upv.es.

T. M. Jones is with the Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, CB3 0FD Cambridge, UK, e-mail: timothy.jones@cl.cam.ac.uk.

This paper is dedicated to the memory of N. Miralaei, who sadly passed away between acceptance of the manuscript and final publication.

The majority of this work was done while A. Valero was with the Department of Computer Engineering, Universitat Politècnica de València.

the cache data array lifetime by reducing the V_{th} degradation, or simply dV_{th}, caused by both phenomena, without incurring performance losses.

This paper makes two main contributions. First, we characterize the cell flips and the duty cycle patterns that high-performance applications cause to each specific memory cell. We find that most applications exhibit regular flip and duty cycle patterns, although they are not always uniformly distributed, which exacerbates the HCI and BTI effects on a small number of cells within the 512-bit cache lines. Results also confirm previous work [9], [10] claiming that most applications write a significant number of near-zero and zero data values into the cache. This behavior has been exploited in the past to address static energy consumption [9] and performance with data compression [10]. Unlike these works, this paper takes advantage of such a behavior to mitigate aging.

Second, based on the previous characterization study, we devise microarchitectural techniques that exploit such a behavior to mitigate aging. The proposal provides a homogeneous degradation of the different cell transistors belonging to the same cache line. For this purpose, the devised techniques aim to reduce cell aging from bit flips and duty cycle and pursue two objectives: i) to spread the bit flips evenly across the memory cells and ii) to balance the duty cycle distribution of the cells. To accomplish the former objective, we propose to progressively shift the bytes of the incoming data lines according to a given rotation shift value that is regularly updated. To attain the latter objective, the mechanism is enhanced to power off those memory cells storing a zero byte value. The result is a switch-off or *sleep* state in which all the cell transistor gate terminals are isolated from electric field stress, thus allowing a partial recovery from BTI [11].

To quantify the benefits of the devised technique, we evaluate the dV_{th} in all the transistors implementing the cache data array, and especially those belonging to the cells with the highest likelihood of failure, that is, those cells showing the highest number of flips and longest duty cycle distribution (either due to logic '0' or '1'). Experimental results show that the savings on the highest number of flips range from 22.4% to 65.5%, whereas the longest '0' and '1' duty cycles can be reduced up to 40.2% and 20.5%, respectively. Finally, the dV_{th} reduction falls in between 21.8% and 44.3%.

The remainder of this paper is organized as follows. Section II provides a short background about BTI and HCI effects in SRAM cells. Section III characterizes the cell flip and duty cycle patterns. Section IV presents the proposed architectural mechanisms. Section V analyzes the experimental results. Section VI summarizes the related work, and finally, conclusions are drawn in Section VII.

II. Background

To help microprocessor architects understand how the logic value (i.e., '0' or '1') distribution among the cache cells as well as the bit flips caused by write operations affect wearout, this section summarizes the implementation of a typical SRAM cell and explains how it suffers from BTI and HCI effects.

As shown in Figure 1, each cached bit is implemented with an SRAM memory cell consisting of 6 transistors (6T). The

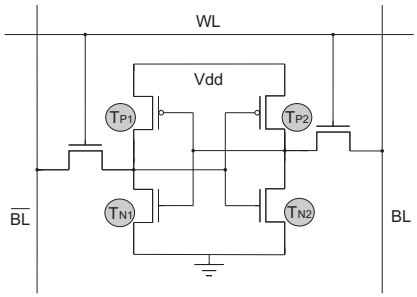


Fig. 1. Implementation of a 6T SRAM cell. The labeled transistors refer to the inverter loop of the cell.

labeled transistors form an inverter loop that holds the stored logic value; this paper uses these labels to refer to these transistors. The remaining pass transistors controlled by the *wordline* (WL) signal allow read and write operations to the cell through the *bitline* (BL) and its complementary ($\overline{\text{BL}}$).

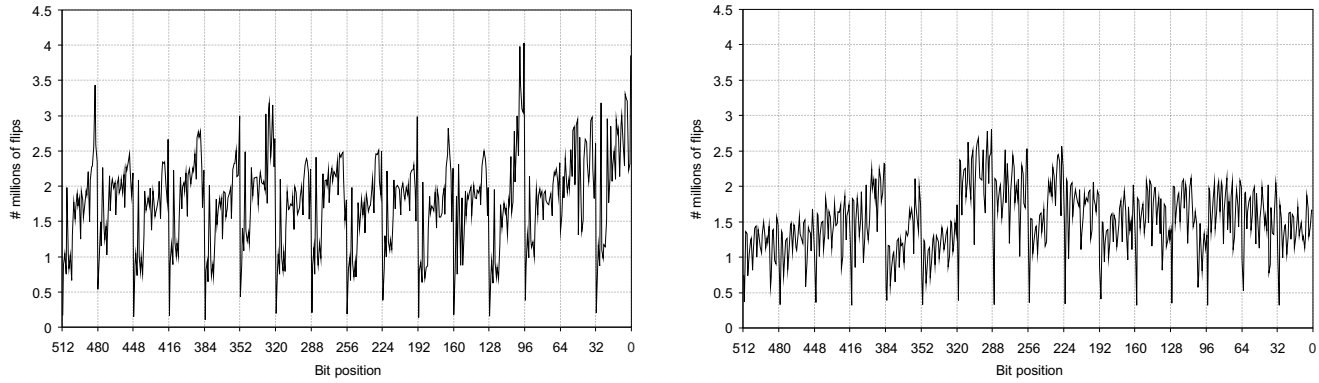
When the SRAM cell is under a '0' duty cycle, that is, when the cell is stable and storing a '0', the PMOS transistor T_{P1} and the NMOS transistor T_{N2} are under stress and they suffer from NBTI and PBTI, respectively. On the contrary, under a '1' duty cycle, transistors T_{P2} and T_{N1} are affected by NBTI and PBTI, respectively. The wearout effects induced by each type of duty cycle are complementary, meaning that, for a given duty cycle, the pair of transistors not under stress are partially under recovery from BTI degradation. Thus, if every cache cell experiences a balanced distribution (i.e., 50%) of '0' and '1' duty cycles, wearout effects due to BTI are minimized and evenly distributed among the inverter loop transistors. Moreover, this reduces the probability of the circuit failing due to Static Noise Margin (SNM) changes.

On the other hand, HCI affects all SRAM cell transistors on a write operation if the logic value flips, regardless of the type of transistor. This effect can be mitigated by avoiding bit flips during write operations. In addition, in order to minimize the chances of SRAM cell faults due to HCI wearout, those remaining bit flips must be evenly distributed among the cells.

To sum up, the inverter loop transistors are continuously aging regardless of whether the cell stores '0' or '1', or is transitioning. This fact makes such transistors particularly sensitive to wearout [7]. Note that the NMOS pass transistors just age when the SRAM cell is being accessed, which represents a very small fraction of the overall execution time, making them much less aging-sensitive than the inverter loop transistors. Thus, this work focuses on wearout mitigation in the inverter transistors only.

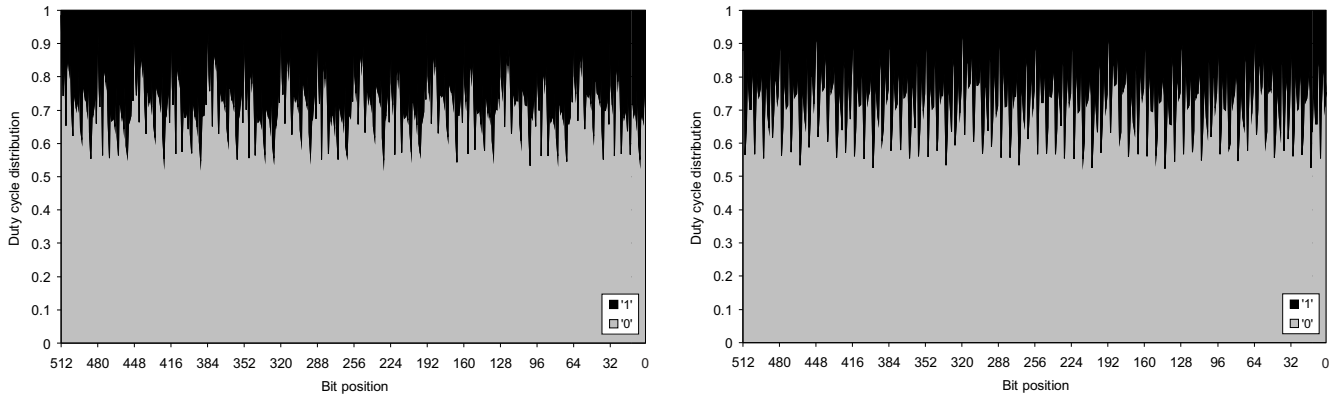
III. Characterization Study

To quantify the impact of both HCI and BTI on the cache memory cells, we have characterized the bit flip and duty cycle patterns, respectively, across the entire SPEC CPU 2006 benchmark suite [12]. For illustration purposes, we show an integer (*perlbench*) and a floating-point (*soplex*) benchmark, since applications of these types use data with different internal representations. Results are shown for the baseline approach, where neither flip nor duty cycle mitigation is employed, on a 64B-line 16KB 4-way L1 data cache in little-endian representation.



a) *perlbench* b) *soplex*

Fig. 2. Number of flips on each bit position of the 64-byte cache lines.



a) *perlbench* b) *soplex*

Fig. 3. Average duty cycle distribution on each bit position of the 64-byte cache lines.

Figure 2 depicts the number of bit flips experienced in each 512-bit cache line. For each bit, the figure plots the sum of the number of flips across all the cache lines. *Perlbench* shows a regular flip pattern across the sixteen 32-bit-word cache lines. Peaks occur in the bits within the Least Significant Byte (LSB) of the words, whereas bits in the MSB account for a much lower number of cell flips. This pattern is common within all integer applications, the main reason being that processors store a significant number of near-zero or narrow integer values to caches [10]. This is mainly due to over-provisioning; that is, programmers usually define relatively large data types (e.g., 4-byte integers) for storing a small value, which could actually be represented with just a few bits. These values are used, for example, to index arrays and matrices, which are extensively accessed inside program loops.

In contrast, the *soplex* floating-point application shows a non-uniform flip pattern due to the IEEE-754 representation of data. Nevertheless, for all the benchmarks, peaks can be also identified in the LSB of some words, resembling the flip pattern of integer applications. This is mainly because floating-point benchmarks also use a significant amount of integer data. For *soplex*, these LSB peaks can be seen in about half of the words (e.g. those starting at bits 224 and 352 of the lines).

Another interesting observation is that the MSB of each word accounts for a much lower number of flips, forming flip dips across the cache lines. This can be clearly seen in *perlbench*, where a large proportion of the stored data are zero and positive near-zero integer values, implying that bits in the MSB hold a logic ‘0’ for a long time. Moreover, the most significant bit of the MSB accounts for a much lower number of flips compared to the remaining bits of the MSB. This is because most integer and floating-point data are positive numbers (sign field set to ‘0’ in the most significant bit of the word). This pattern is common for all the remaining SPEC2006 benchmarks.

Figure 3 shows the average duty cycle distribution across the bit positions of all the cache lines. As observed, logic ‘0’ is the predominant value, thus reinforcing the intuition behind Figure 2 results. Unlike Figure 2 though, Figure 3 shows the fraction of time that each logic value is stored. In general, ‘0’ is stored for longer than ‘1’ because memory is usually initialized to zero when it is allocated. Thus, even if there was an equal likelihood of an application writing a ‘0’ or ‘1’ in any bit position, this initialization will always mean ‘0’ is stored for longer. Other reasons for ‘0’ being stored longer are that false boolean values and NULL pointers are represented with zero, as well as most data in dense-form sparse matrices [10].

Notice too that those ‘0’ duty cycle peaks close to 0.9 (i.e.,

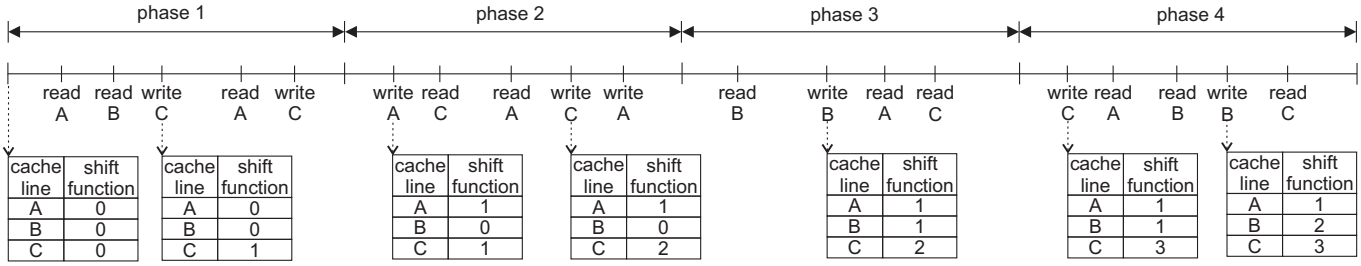


Fig. 4. Time diagram with the execution time divided into phases and the associated shift function transitions for the cache lines *A*, *B*, and *C*.

90% of the time) that are encountered in both benchmarks correspond to the flip dips displayed in Figure 2. This behavior is similar for all the remaining benchmarks. In addition, it can also be seen that *perlbench* shows a wider zero duty cycle peak in comparison to *soplex*, meaning that the former presents longer execution periods with '0' values stored in the MSB.

IV. Proposed Architectural Techniques

Based on the characterization study, this work proposes to enhance the L1 data cache design to mitigate both HCI and BTI wearout. First, the HCI effect is attacked by spreading out the bit flips across each word's memory cells. Second, BTI is minimized by powering off cells when a zero byte is fetched into them. Finally, both mechanisms are evaluated working together in a joint cache design in terms of circuitry, area, and timing overhead.

The proposed approaches attack aging in the data array. Given that the tag array is much smaller than the data array, resilient technologies could be used to address tag wearout. For example, resilient 8T cells introduce a 19% area overhead compared to typical 6T cells [13]. According to CACTI [14], implementing the tags with 8T cells results in just a 1.95% area overhead for a 16KB L1 cache. However, further consideration of tag aging is beyond the scope of this paper.

A. HCI Mitigation: The BW Approach

The previous study showed that the memory cells containing the LSB of a word experience bit flip peaks across all these cells in a cache line, especially for integer applications. These LSB cells age the most from HCI because of the direct relationship between HCI and flip activity.

To mitigate HCI wearout, our initial scheme implements a rotation shift mechanism that distributes flips located on the peaks across several bit positions within the words. In particular, the proposed technique, named BW (Bytes-within-Words shifting), periodically shifts the bytes within the words on a round-robin basis. BW uses 4 *x-shift* functions for a 4-byte word, where *x* means the number of bytes to be shifted. For instance, a 1-shift function performs a rotation shift from the LSB to the MSB by one byte. Therefore, after the shift operation, the contents of the LSB are those that were previously stored in the MSB. Note that other shift mechanisms like bits-within-words shifting could bring higher flip peak savings; however, the required hardware to implement these approaches would have a larger impact on area and performance (see Section IV-C).

To make hardware simple, all the 16 words in the cache line follow the same shift function at any given time. Thus, each cache line simply requires one pair of control bits, namely C_{BW0} and C_{BW1}, to keep track of the current shift. When the line is read, these bits are used to realign the word bytes and forward them with their correct positions to the processor.

An important design issue is the length of time a shift function should be maintained before applying the subsequent shift. Figure 4 shows a working example to illustrate how the shift functions are updated. The execution time is divided into phases with a fixed number of processor cycles. Initially, the cache lines *A*, *B*, and *C* follow a 0-shift function, that is, the stored words have not experienced any byte shift. A shift transition occurs the first time that a cache line is written during a phase. This occurs for line *C* within the first phase, where the mentioned write operation induces a transition from 0-shift to 1-shift. The updated shift function remains valid for all subsequent accesses to line *C* during the remainder of the current phase (i.e., the second write operation to *C* within the first phase does not modify the shift function). Reads in any phase do not update the shift function. The current shift function remains in force until a new write operation occurs in a later phase (i.e., write to *C* in the second phase). As in the first phase, the second phase updates the shift function of lines *A* and *C* upon their first write, while during phase 3 the only line whose shift function is modified is line *B*. The shift transitions are managed by adding a single control bit per cache line, referred to as C_{pBW}, to indicate whether the line has been ever written in the phase or not. Of course, the C_{pBW} bit is reset every time a new phase starts. Notice that, at any given time, different cache lines can follow distinct shift functions since writes do not act in a synchronized way. Moreover, usually only a small number of cache lines are accessed at a given point in time.

Finally, experiments showed that a phase length of 8M processor cycles noticeably reduces the high flip peaks. Very large phases result in fewer shifts and longer times using each shift function, which had only a minor effect on reducing those high flip peaks. On the contrary, *over-shifting* with very short phases significantly increased the overall number of flips, leading to high peaks. For instance, compared to an 8M phase length, 4M-cycle and 16M-cycle phases increase the highest flip peak on average by 4.4% and 3.8%, respectively, for the studied benchmarks.

B. BTI Mitigation: The SZB Mechanism

The duty cycle distribution (see Section III) confirmed that current applications keep a high number of cache line bits as '0', which accelerates BTI wearout in the SRAM

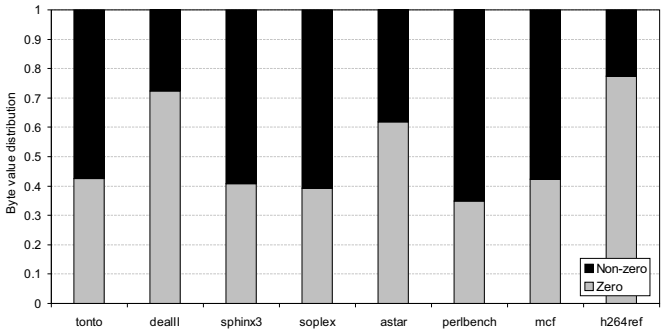


Fig. 5. L1 data cache byte value distribution split into zero and non-zero for the studied applications.

cell transistors T_{P1} and T_{N2} from Figure 1. For the sake of completeness, we have obtained the percentage of zero bytes in the L1 data cache. Figure 5 shows the results for the 8 SPEC2006 applications (4 integer and 4 floating-point) that most stress the L1 data cache. As observed, all of these applications keep bytes equal to zero more than 30% of the time. Moreover, in *dealII*, *astar*, and *h264ref*, this percentage exceeds 60% of the time. Overall, the percentage of zero bytes is on average by 51%[1].

Based on these results, simply skipping the writing of zero byte values to the cache and maintaining the previously cached byte could reduce the amount of time that the memory cells contain a logic '0' value [15]. However, this may imply storing '1' for longer periods of time, which would speed up BTI in transistors T_{P2} and T_{N1}.

To combat the BTI phenomenon in all transistors at the same time, we introduce the Switch-off Zero Byte (SZB) mechanism, which switches off those byte cells that store a zero value. Powering off an SRAM cell implies that all the SRAM cell transistors are partially under recovery from BTI [11]. Moreover, notice that by turning off zero bytes, the temperature in the cache would decrease, which would also help mitigate the cache aging. Similarly, the overall cache energy consumption is also reduced (see Section V-E).

The SZB technique works as follows. On a cache miss or a write hit, each byte to be written to the cache is compared to zero. If the comparison matches, a control bit per byte, called C_{SZB}, is set to indicate that the associated byte stores a zero, and the byte memory cells are powered off. On a cache read hit, the corresponding C_{SZB} control bits for the target line are checked, and if any of them is set, a zero byte is forwarded to the processor since the relevant data array cells are not active.

C. Hardware Implementation and Operation

This section discusses a possible basic hardware implementation of the proposed joint design. The main focus of the paper is not to deal with the optimal implementation but on providing some insights on the design. Further enhancements could be provided with an optimized design, which is beyond the scope of this work.

[1]Compared to the byte granularity, the percentage of zero words over the total number of words is reduced on average down to 32%, meaning that the effectiveness of the SZB mechanism diminishes.

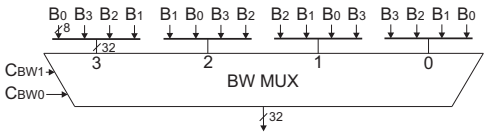


Fig. 6. Write circuit for the BW mechanism.

In addition, this section also evaluates how to deal with the HCI and BTI effects in the control bits, and how the read and write operations are performed in the proposed cache design, including timing issues.

1) **Hardware Components and Area Overhead:** The BW mechanism can be implemented with 16 4-to-1 multiplexers; one for each data word within the incoming line. Figure 6 shows one of the multiplexers and its associated inputs used in the write circuit. Label B_i refers to the different data bytes from the word, B_0 and B_3 being the LSB and MSB, respectively. Each data input consists of the data bytes ordered according to one of the 4 possible shift functions. The multiplexer is controlled by the C_{BW0} and C_{BW1} control bits that correspond to the current shift function. For the read circuit, another 16 4-to-1 multiplexers can be used for the requested line; however, the order of the data inputs differs from those of the write circuit, since in this operation the contents must be realigned instead of shifted. These multiplexers are only used when reading and writing a given line, thus, they are shared among all the lines in the data array.

We have modeled these multiplexers using CACTI, which reports an area of $9.009\mu m^2$ for each of them using a 32nm technology node. Considering the whole cache data array area, that is $0.226mm^2$, the overall area overhead of the 32 BW multiplexers is just 0.13% of the data array area. Recall that the BW approach requires 3 bits for each cache line to hold the control bits, which translates into an overhead of 768 bits for the studied 16KB L1 data cache. Such a storage overhead is just 0.59% of the cache data array capacity.

For the proposed SZB mechanism, simple hardware is required to compute the C_{SZB} bits and to forward zero bytes [9]. Figures 7(a) and 7(b) show the required circuitry for a cache word. On a write operation, the C_{SZB} bits are set by *ORing* the data bits of each byte, thus a control bit set to '0' indicates that the associated byte is zero. On a read operation, tristate buffers driven by the control bits are used to forward either zero byte values, or the actual written byte from the cache.

The switch-off mechanism is implemented using the gated-V_{dd} technique [16], which has been widely adopted in cache designs like Cache Decay [17] for leakage energy mitigation. In particular, our design includes a PMOS and an NMOS sleep transistor that connect the SRAM cell to V_{dd} and ground, respectively. This power-gating configuration completely isolates both cell nodes, placing them into a switch-off state when the sleep transistors are disabled, preventing the nodes from storing a given logic value [11]. Otherwise, disabling either the V_{dd} or ground path exclusively results in both nodes holding a '0' or '1', respectively, which could speed up BTI aging [18]. Contrary to the cell transistors, the sleep transistors are implemented using high-V_t devices to make them resilient against BTI and HCI [19], [20].

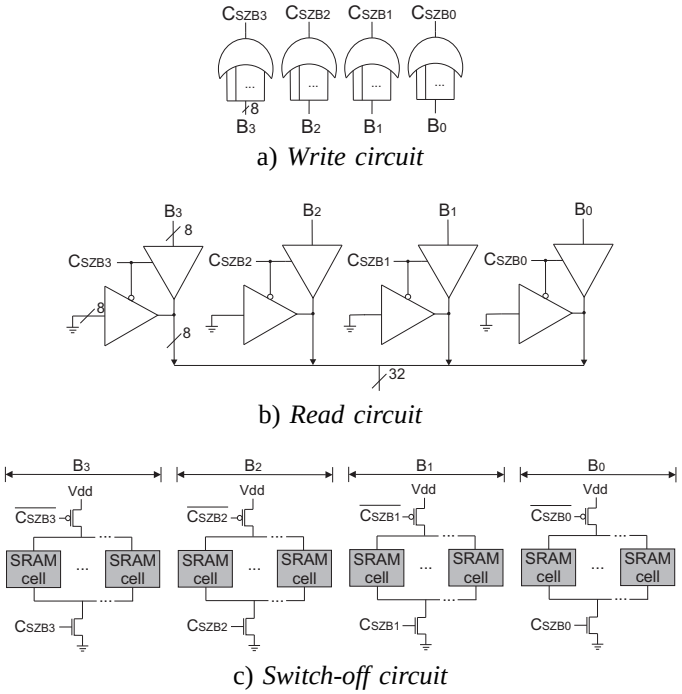


Fig. 7. Read/write and switch-off circuits for the SZB approach.

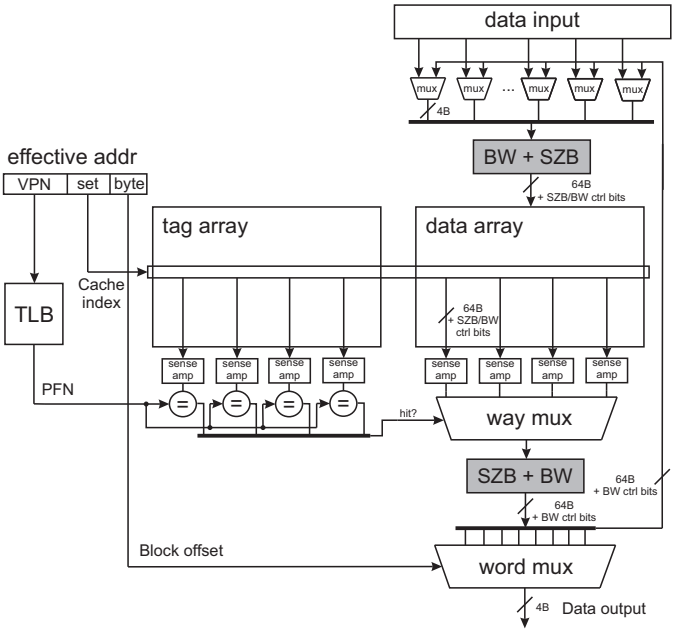


Fig. 8. Block diagram of the L1 data cache access including the proposed components (grey boxes).

The gated-V_{dd} technique can be applied at different levels of granularity [16]. For simplicity and given that the proposed SZB technique works at byte granularity, a pair of sleep transistors (each in the ground and V_{dd} paths) is enough to cut off those memory cells storing a data byte as depicted in Figure 7(c). Both sleep transistors are controlled by the corresponding C_{SZB} control bit. When the C_{SZB} bit is '0' (i.e., the incoming byte is zero), the transistor gates are disabled and the cells are switched off. Otherwise, the cells are powered on to store the fetched data. Finally, notice that the simple SZB circuitry consisting of control bits, sleep transistors, and remaining read/write hardware has a low area overhead (11.76% of the 16KB L1 cache) [9].

2) **Control Bit Inversion:** Both HCI and BTI phenomena should be evaluated not only in the data array bits but also in the additional control bits added by our mechanisms and implemented as SRAM cells. Recall that the C_{BW0} and C_{BW1} bits make up a 2-bit counter and they are updated between regular shift phases of 8M processor cycles, which results in an implicit balanced (i.e., *near-optimal*) duty cycle distribution in such bits. However, the C_{pBW} bit is set to '1' when the associated line is written for the first time within a phase, and set to '0' every time a new phase starts. We have evaluated that such writes normally come soon after the phase begins, causing a highly-biased '1' duty cycle in these bits, which exacerbates BTI in transistors T_{P2}-T_{N1}.

To deal with this drawback, we periodically complement all the C_{pBW} bits between shift phases, which allows us to achieve a near-optimal duty cycle ratio. Similar to the C_{pBW} bits, the C_{SZB} bits are also inverted between shift phases, since some applications store zero (long '0' duty cycle in C_{SZB}) and non-zero (long '1' duty cycle) bytes in the same location for an extended period of time. Please refer to Section V-B and Section V-C for further details. Note that to recover the original logic value of the control bit, the implementation only requires a simple 2-input XOR gate whose inputs are: i) the value currently stored in the control bit and ii) a bit identifying if the current phase number is either odd or even (i.e., if the stored control bit must be inverted or not).

Of course, such an inversion of the control bits between shift phases brings an increase of the number of flips, which could speed up the HCI degradation. Nevertheless, as the phase length is large enough, this increase does not have a significant impact as evaluated in Section V-A.

3) **Read/Write Operations:** With the aim to clarify how both BW and SZB schemes work together, Figure 8 plots a cache block diagram with both mechanisms represented as grey boxes. On a cache read hit, after the way multiplexer selects the target line from the selected set, its contents and the associated control bits are forwarded to the SZB read circuit. Once the SZB tristate buffers have forwarded the zero bytes, the BW multiplexers realign the bytes and serve the original line to the processor. Note that, on a read operation, there is no need to restore the power to those memory cells that originally would hold zero bytes.

On a write hit, the contents stored in the target line are read and forwarded to the upper-side multiplexers, which compose the line to be stored jointly with the input data. Meanwhile, the cells of the target line are powered on (if any). Then, the BW write circuit multiplexers rotate the bytes of each word in the line according to the corresponding shift function. After that, the SZB write circuit computes the SZB control bits, prevents the zero bytes from being written into the data array, and switches off the corresponding cells. On a cache miss, the same circuitry is used to store the incoming data.

The way multiplexer is the only one that muxes the C_{SZB} and C_{BW} control bits from each line, which are used as control entry for the proposed SZB read circuit and BW read/write

TABLE I
ARCHITECTURAL MACHINE PARAMETERS.

Microprocessor core	
Processor frequency	3GHz
Issue policy	Out of order
Fetch, issue, commit width	4 instructions/cycle
ROB size (entries)	256
# Int/FP ALUs	4/4
Memory hierarchy	
L1 data & insn caches	16KB, 4-way, 64B-line, 1-cycle tag array, 3-cycle data array
L2 unified cache	256KB, 8-way, 64B-line, 6-cycle tag array, 10-cycle data array
L3 unified cache	4MB, 16-way, 64B-line, 11-cycle tag array, 23-cycle data array
Main Memory	200-cycle

circuits, respectively. According to CACTI, the area overhead of muxing these control bits is only 0.09% of the data array.

4) **Timing:** To study the impact on the cache access time, we have considered the delays involved. The write operation constitutes the largest path, since the target line is first read and then written to the cache. CACTI reports a 0.765ns access time for such a write operation. The delay of each 4-to-1 multiplexer used to implement the shift functions is 0.088ns, whereas muxing the control bits adds a delay of 0.044ns. The delay of the SZB OR gates and tristate buffers can be assumed to be negligible [9].

Overall, the write access time becomes 0.985ns taking into account the additional circuitry. For the assumed 3GHz processor, therefore, the difference (in ns) over the original delay is masked when the access time is quantified in processor cycles. That is, the additional circuitry has no impact on the data array access time (in cycles), although it could impact on other processor designs (e.g., those working at a higher frequency). In this case, an optimized or alternative design for the additional circuitry would be required. Notice too that this additional delay will be amortized over time compared to the perpetual delay caused by the studied wearout effects.

Finally, note that in the write operation, the delay of turning an SRAM cell on or off just involves the switching delay of the sleep transistors (which is only 4.2ps according to HSpice using 32nm PTM models [21] and ITRS 2011 documents [22]) and the delay to stabilize the cell when powering on. This minor delay is entirely masked while the read data are traversing the BW+SZB circuits and the input data multiplexers. Further, the power on/off mechanism is not used in the read operation, which actually corresponds to the critical path of the cache access, thus the read access time is not negatively impacted.

V. EXPERIMENTAL RESULTS

We extended the Multi2Sim simulation framework [23] to implement both BW and SZB approaches. Experiments were performed for the 32-bit x86 ISA with the *ref* input set, while results were collected simulating 500M instructions after skipping the initial 500M instructions. Table I summarizes the main architectural parameters. All the cache access times were obtained from CACTI for a 3GHz processor clock and a 32nm technology node.

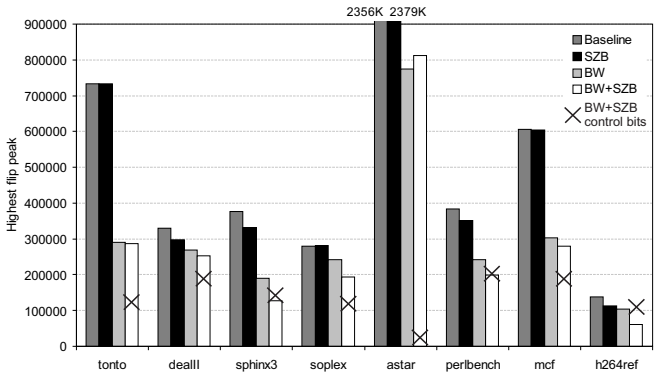


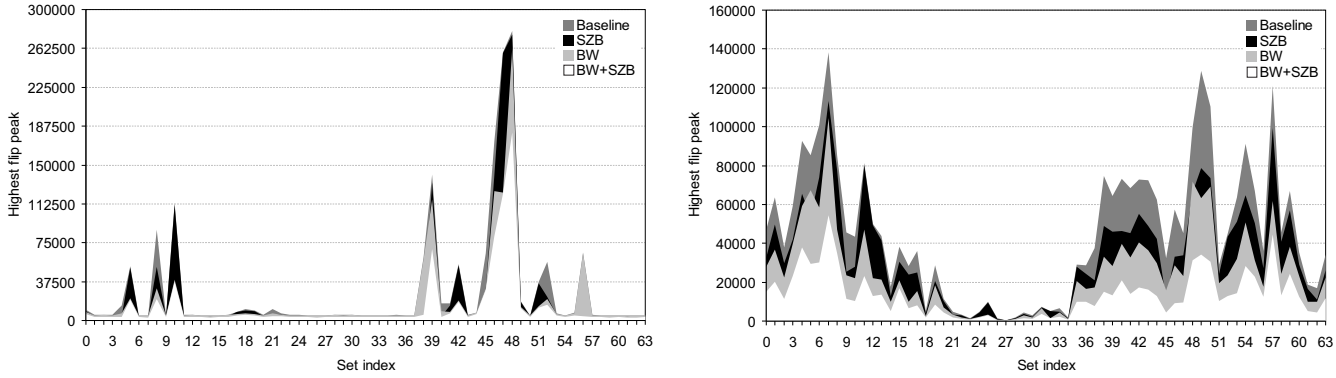
Fig. 9. Highest cache flip peak across all the studied applications.

A. Cell Flip Analysis

With the aim to provide insights about the HCI wearout reduction brought by our proposed mechanisms, this section identifies the maximum cache flip peak across all the analyzed benchmarks, classifies applications according to their flip peak behavior, and quantifies the number of flips the mechanisms save. Figure 9 plots the raw number of flips for the cache memory cell that holds the highest flip peak for the baseline scheme, SZB and BW working alone, and both schemes working together (BW+SZB).

For all the studied benchmarks, the HCI-aware BW mechanism substantially reduces the number of flips with respect to the baseline scheme by shifting the incoming data. On the other hand, the benefits brought by SZB are less remarkable. In fact, SZB slightly increases the number of flips on the highest peak in *soplex* and *astar*. This is due to a write operation on a decayed byte triggering the switch on of the memory cells, which in turn implies the appearance of a random logic value before writing the input values. Transitioning from a random value instead of from the data actually stored (i.e., a zero byte in the baseline scheme) can induce a higher number of flips in a given cell. These transitions actually save flips in the highest peak compared to the baseline for the remaining benchmarks. Moreover, by combining both mechanisms (BW+SZB), the overall effect is to save more flips than BW in all the studied applications apart from *astar*.

The figure also shows the highest flip peak in the additional control bits used by BW+SZB, represented with a cross symbol, and accounts for C_{SZB}, C_{pBW}, C_{BW0} and C_{BW1} as introduced in Section IV. For both C_{SZB} and C_{pBW} bits, results include the flip overhead due to inverting them between shift phases. For all the studied benchmarks, the maximum flip peak rises in the C_{SZB} bits since their flip activity is much higher than in those of the BW mechanism, which can only flip once in a large phase of 8M processor cycles (see Section IV-A). Nevertheless, for most applications (5 out of 8) the BW+SZB data array has a higher flip peak than the C_{SZB} bits. This is because flips in such control bits only occur due to byte write sequences in the same byte location with a non-zero value followed by a zero value and vice versa, whereas all sequences



a) *soplex* b) *h264ref*

Fig. 10. Highest flip peak per cache set for a subset of the analyzed benchmarks.

with a non-zero value followed by a distinct non-zero value induce cell flips in the data array. In *h264ref*, the C_{SZB} peak is relatively high because this application writes a large number of zero bytes (see Figure 5) and most sequences are those affecting the C_{SZB} bits. Still, the C_{SZB} peak is always much lower than that of the baseline data array for all the studied applications. Overall, the BW+SZB flip savings range from 22.4% (*h264ref*) to 65.5% (*astar*).

Figure 10 depicts the raw number of flips for the memory cell within each cache set that holds the highest flip peak for *soplex*, which is a representative benchmark for those applications with a higher data array flip peak in SZB than in the baseline (see Figure 9), and *h264ref*, which is representative for those that reduce the SZB flip peak over the baseline.

The highest peak value widely varies across the cache sets due to the non-uniform distribution of accesses across them. For *soplex*, the highest cache flip peaks are always located in the set with index 48 regardless of the studied mechanism. Like in this set, the baseline and SZB schemes have a similar flip peak in sets 5, 10, 39, and 42, whereas BW+SZB largely reduces the amount of flips in all of them. For *h264ref*, the cache set 7 contains the highest peaks for all the analyzed mechanisms. In this case, it can be appreciated that BW+SZB is the scheme that most saves the highest flip peak in all the cache sets, followed by BW, SZB, and the baseline.

B. '0' Duty Cycle Analysis

This section provides insights about the BTI wearout savings brought by the proposed techniques by quantifying the maximum duty cycle distribution when storing a logic '0' value. Figure 11 plots the longest '0' duty cycle distribution for the whole cache across the studied benchmarks. The duty cycle is split into '0' and '1' duty cycle for the baseline and BW approaches, while SZB and BW+SZB incorporate the switch-off state, which refers to the amount of time that cells spend powered off.

The maximum '0' duty cycle for the baseline scheme is nearly 100% in all the analyzed benchmarks, meaning that at least one memory cell within the cache contains a '0' for the majority of the execution time. In contrast, by rotating the bytes of the fetched words, not only flips are distributed

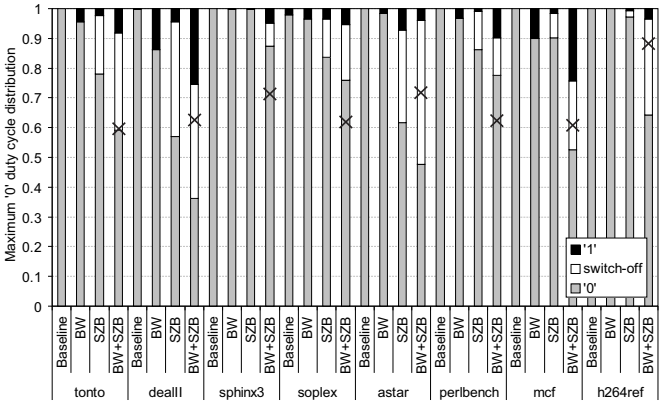


Fig. 11. Maximum '0' duty cycle distribution on the entire cache across all the studied applications. The cross symbol refers to the longest '0' duty cycle in the control bits.

across the memory cells but also the logic '1' values. This helps obtain a modest '0' duty cycle saving compared to the baseline. Larger savings come when using the BTI-aware SZB approach, which reduces the '0' duty cycle according to the amount of zero bytes that account for the switch-off state. Finally, by combining both proposed techniques, results are enhanced for a much lowered '0' duty cycle, even mitigating the '0' duty cycle beyond the balanced 50% ratio in applications like *dealII* and *astar*.

The cross symbol in the graph indicates the longest '0' duty cycle distribution in the control bits of the BW+SZB approach. Similarly to the cell flip analysis, the maximum '0' duty cycle in these bits is given by those of the SZB approach. This is due to the C_{pBW} control bits holding a logic '1' during most of the program execution time (see Section IV-C2), and by periodically inverting these bits between shift phases, their maximum duty cycle becomes balanced (between 40% and 50% in most benchmarks). In contrast, for the C_{SZB} bits, even with the bit inversion, the duty cycle does not approach the optimal balance as much as the C_{pBW} bits do because of the existing variability of the stored logic values in the C_{SZB} bits across the different shift phases. Nevertheless, the C_{SZB} longest '0' duty cycle is relatively close to 50% with the only exception being *h264ref*. In addition, the C_{SZB} duty cycle is

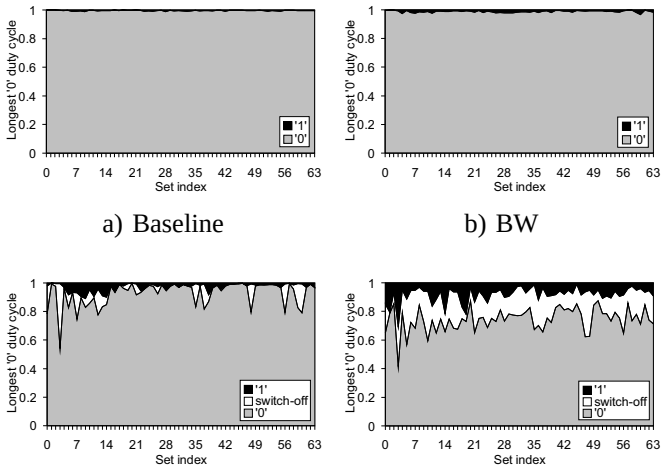


Fig. 12. Longest ‘0’ duty cycle distribution per cache set for the studied mechanisms in *sphinx3*.

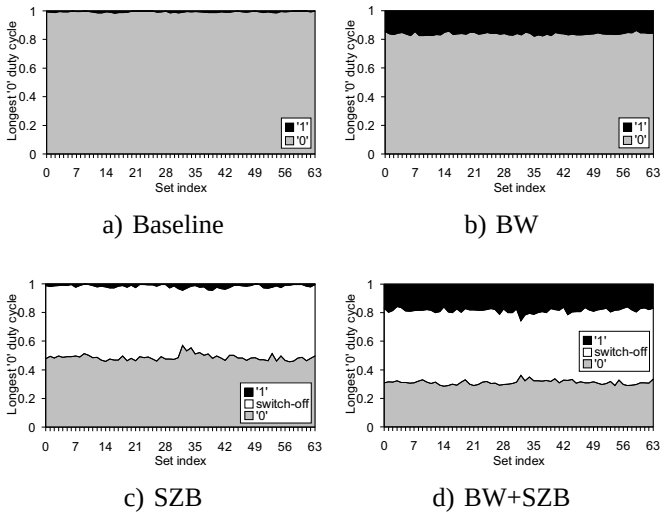


Fig. 13. Longest ‘0’ duty cycle distribution per cache set for the studied mechanisms in *dealII*.

always lower than that of the baseline data array. Overall, the BW+SZB ‘0’ duty cycle reduction falls in between 11.8% (*h264ref*) and 40.2% (*tonto*).

Figure 12 and Figure 13 illustrate the longest ‘0’ duty cycle distribution per cache set in the *sphinx3* and *dealII* applications, which are those benchmarks that experience the least and most reductions in the ‘0’ duty cycle in the data array, respectively, for the BW+SZB approach. Results show that the baseline obtains a near 100% ‘0’ duty cycle in all the cache sets for both applications. Note that, for BW, shifting the incoming bytes does not benefit any set for *sphinx3*, while the ‘0’ duty cycle reduction for *dealII* is quite low. On the other hand, SZB brings larger savings with the inclusion of the switch-off state. However, this is not enough for *sphinx3*, where a long ‘0’ duty cycle is still present in most cache sets. In contrast, *dealII* substantially saves the ‘0’ duty cycle thanks to its high number of zero bytes (see Figure 5) and the large amount of time that they are cached. Finally, BW+SZB

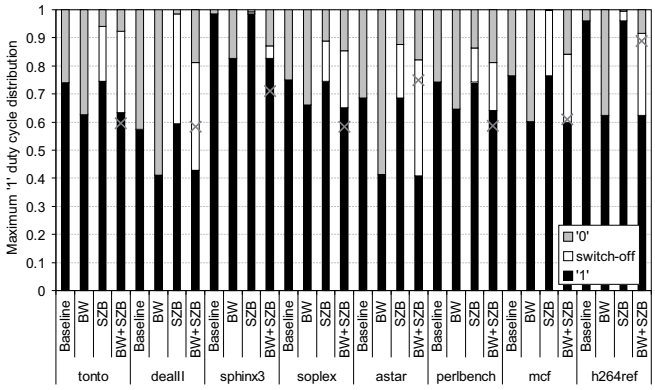


Fig. 14. Maximum ‘1’ duty cycle distribution on the entire cache across all the studied applications. The cross symbol refers to the longest ‘1’ duty cycle in the control bits.

achieves larger ‘0’ duty cycle reductions not only from the switch-off state but also from spreading such a state across the memory cells for both benchmarks.

C. ‘1’ Duty Cycle Analysis

The proposed SZB and BW+SZB techniques mitigate the BTI effect in the pair of SRAM cell transistors T_{P1}-T_{N2} by switching off those cells holding a zero byte. However, for a complete evaluation of the BTI phenomenon, the longest ‘1’ duty cycle distribution should be analyzed to provide insights about the BTI reduction in transistors T_{P2}-T_{N1}.

Figure 14 illustrates the maximum ‘1’ duty cycle for the entire cache across the studied benchmarks. As expected, the ‘1’ duty cycle is not as critical as its counterpart ‘0’ duty cycle since most stored data bits are ‘0’. Similar to the previous analysis, the BW technique enhances the duty cycle distribution with respect to the baseline by shifting logic ‘1’ values across the memory cells of the lines. The SZB approach by itself is not able to mitigate the longest ‘1’ duty cycle with respect to the baseline approach. This is because SZB attacks the ‘0’ duty cycle by switching off zero bytes; however, the amount of time that a ‘1’ is stored in a given cell remains unchanged. The same reasoning can be made when comparing the maximum ‘1’ duty cycle of BW and BW+SZB.

As above, the cross symbols refer to the longest ‘1’ duty cycle for the BW+SZB control bits, which is given by the C_{SZB} bits. Compared to the baseline longest ‘1’ duty cycle in the data array, only 2 out of 8 applications (*dealII* and *astar*) present a slightly higher C_{SZB} ‘1’ duty cycle distribution. Taking into account the control bits, the BW+SZB ‘1’ duty cycle distribution savings are up to 20.5% (*mcf*).

Figure 15 and Figure 16 show the maximum ‘1’ duty cycle distribution per cache set for the studied mechanisms in *sphinx3* and *dealII* applications, respectively. As can be seen, the baseline approach shows a highly-biased ‘1’ duty cycle in *sphinx3*, especially in those sets with a low index. On the contrary, this approach obtains a balanced duty cycle distribution in *dealII*, confirming the high amount of zero data values on this benchmark. Compared to the baseline, BW reduces the ‘1’ duty cycle ratio in all the sets of the analyzed benchmarks. Results also show the ‘1’ duty cycle similarity in

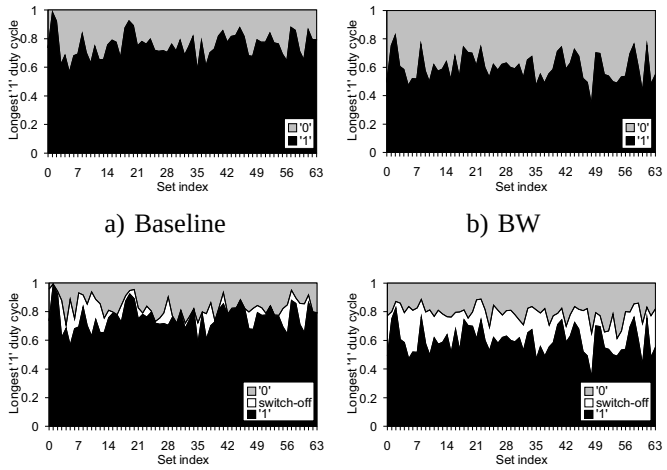


Fig. 15. Longest ‘1’ duty cycle distribution per cache set for the studied mechanisms in *sphinx3*.

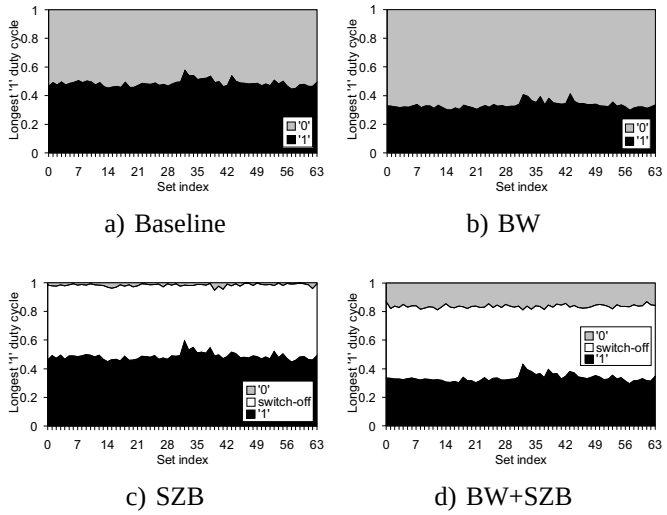


Fig. 16. Longest ‘1’ duty cycle distribution per cache set for the studied mechanisms in *dealII*.

all the sets when comparing the baseline with SZB and BW with BW+SZB, which corroborates the results presented in the preceding analysis.

Finally, comparing the maximum ‘1’ duty cycle results with those of the ‘0’ duty cycle (see Figure 11), we can conclude that, for most benchmarks and mechanisms, the ‘0’ duty cycle will induce a higher BTI wearout in the corresponding SRAM cells since its maximum distributions are larger than those of the ‘1’ duty cycle. Such SRAM cells will fail sooner than the others, and will have much more NBTI and PBTI degradation in transistors T_{P1} and T_{N2}, respectively.

D. Comparison Against the Colt Mechanism

Among the state-of-the-art approaches that attack cache aging, we compare BW+SZB to the Colt mechanism [8] since, to the best of our knowledge, it is the only one that addresses both HCI and BTI effects at the same time.

Colt proposes two different techniques. First, the duty cycle is balanced by splitting the execution time into epochs of

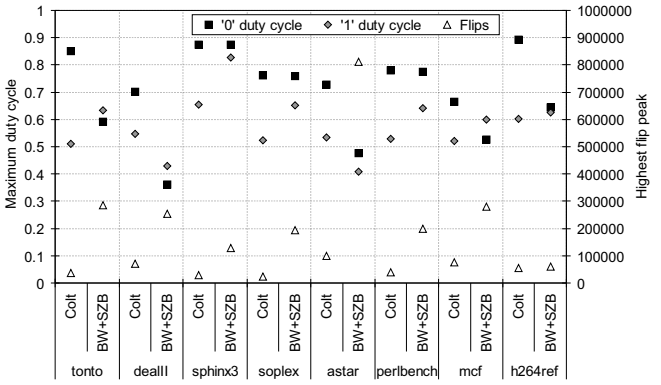


Fig. 17. Highest flip peak and maximum ‘0’ and ‘1’ duty cycles for Colt and BW+SZB mechanisms across all the studied applications.

1ms and alternatively writing the incoming data from L2 as normal or complemented within each epoch. The second technique minimizes flip peaks by uniformly spreading the cache accesses across sets. To do so, the cache index function is altered with an LFSR, which is updated on every change of epoch. However, the cache contents must be flushed when a given epoch finishes, which incurs in performance degradation and energy overhead with respect to the baseline cache design.

Figure 17 plots the highest flip peak and longest ‘0’ and ‘1’ duty cycles for Colt and the proposed BW+SZB technique. Compared to BW+SZB, Colt will mitigate HCI wearout by reducing the highest flip peaks. Like BW+SZB, the ‘0’ duty cycle is much more remarkable than its counterpart ‘1’ duty cycle, with differences by 0.3 in applications like *tonto* and *h264ref*. More importantly for total aging (see Section V-F3), the ‘0’ duty cycle exceeds the longest duty cycle obtained by BW+SZB in all the studied applications, meaning that Colt will induce a higher BTI degradation than BW+SZB. This is mainly due to Colt just complements the written data coming from the L2 cache, whereas L1 write hits store the data non-complemented. Thus, those write hits occurring right after a cache miss nullify the complemented effect, leading to memory cells storing logic ‘0’ for a large amount of time.

E. Energy Savings

The proposed SZB mechanism saves energy thanks to powering off zero bytes for aging purposes. This section analyzes the data array energy consumption of the baseline and BW+SZB approaches. The energy overhead of the additional BW+SZB circuitry has been taken into account by modelling these components with CACTI, which calculates cache leakage and dynamic energy expenses. These expenses were combined with the processor statistics from Multi2Sim to obtain the overall consumption.

Figure 18 depicts the normalized results. Regardless of the analyzed approach, leakage and dynamic expenses significantly differ among benchmarks since they mainly depend on the execution time and number of cache accesses, respectively. Thus, applications enlarging the execution time (e.g., *mcf*) or increasing the number of accesses (e.g., *h264ref*) present a larger contribution of each type of energy. Compared to the baseline, the BW+SZB scheme saves leakage by switching

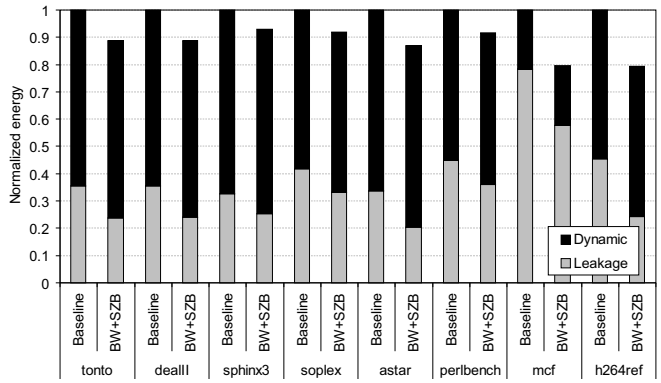


Fig. 18. Normalized consumption for the baseline and BW+SZB mechanisms.

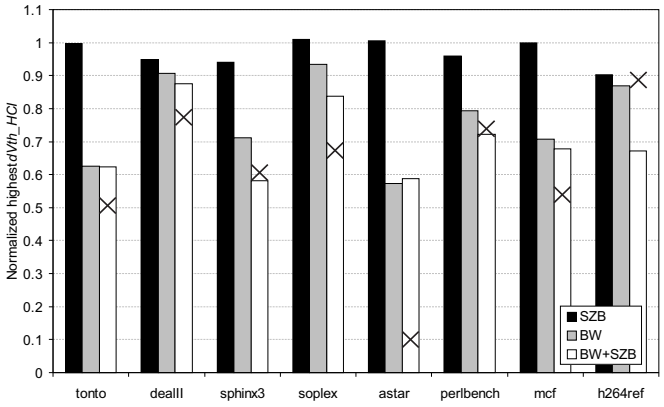


Fig. 19. Normalized highest dV_{th} caused by the HCI effect on the entire cache. The cross symbol refers to the normalized highest dV_{th}_HCI in the control bits.

off zero bytes. These savings largely compensate the leakage incurred by the additional BW+SZB circuitry. On the other hand, BW+SZB slightly increases the dynamic energy with respect to the baseline due to the additional logic. Nevertheless, the overall effect is to mitigate the total consumption in all the studied benchmarks, with savings ranging from 7.1% (*sphinx3*) to 20.6% (*h264ref*).

F. Estimation of the dV_{th} Savings

This section analyzes the V_{th} degradation caused by the analyzed aging effects. Results are shown for the SRAM cell transistor with the highest dV_{th} (not necessarily the same transistor across the studied approaches), which is the one that suffers the highest wearout.

The dV_{th} has been obtained assuming a 3-year lifetime for our 32nm technology node [24]. To complete this execution period, we assumed that the benchmark execution is repeated over and over until the established lifetime is reached [25]. First, we focus on the dV_{th} caused separately by the HCI (dV_{th}_HCI) and BTI (dV_{th}_BTI) effects. Then both phenomena are evaluated jointly.

1) V_{th} **Degradation from HCI:** The dV_{th}_HCI results were calculated using the standard dV_{th} formula presented in Equation 1 [25]. All the parameters are constant values apart from t, which refers to the amount of time (in seconds) that the memory cells flip their contents. The reader is referred to [25] for further information about the constant parameters.

$$\Delta V_{th\_HCI} = A_{HCI} \times \alpha \times f \times e^{\frac{V_{dd}-V_t}{t_{ox} \times E_{HCI}}} \times \sqrt{t} \tag{1}$$

Figure 19 plots the normalized highest dV_{th}_HCI with respect to the baseline technique. Remember that HCI affects uniformly all the SRAM cell transistors. Thus, the presented results are those from the memory cell with the highest HCI wearout. As expected from Equation 1, the dV_{th}_HCI is highly related to the number of flips shown in Figure 9, confirming that the proposed BW+SZB mechanism effectively mitigates the HCI degradation. Results also corroborate that the dV_{th}_HCI in the additional control bits is always lower than that of the baseline data array.

2) V_{th} **Degradation from BTI:** The dV_{th}_BTI results were computed from the standard formula shown in Equation 2 [21]. All the parameters are constant values except *t_stress* and *t_rec*, which refer to the amount of time (in seconds) that the cell transistors are under stress and recovery modes, respectively. Please refer to prior work [21] for further details.

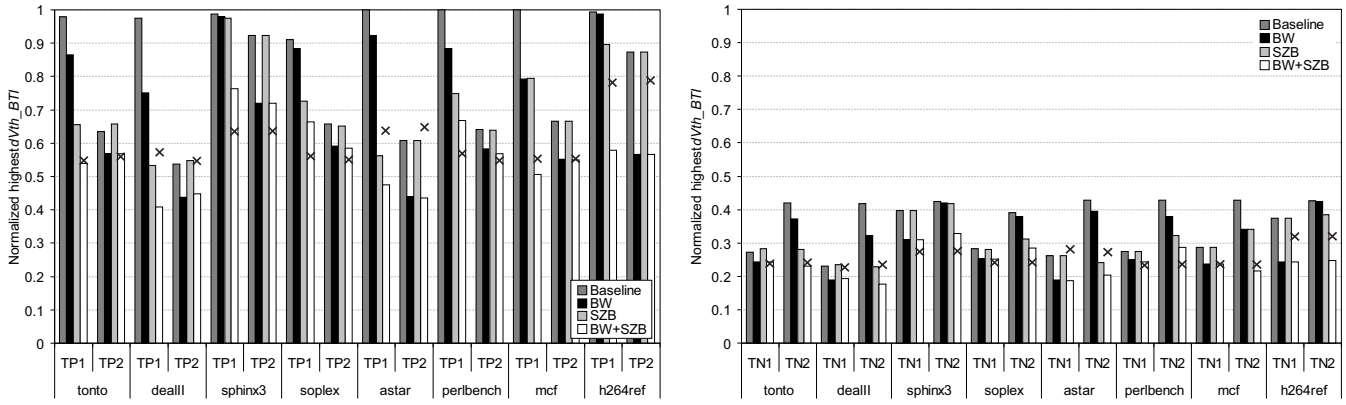
$$\begin{aligned} \Delta V_{th\_BTI} = A_{BTI} \times t_{ox} \times \sqrt{C_{ox} \times (V_{dd} - V_t)} \times \\ (1 - \frac{V_{ds}}{\alpha \times (V_{dd} - V_t)}) \times e^{\frac{V_{dd}}{t_{ox} \times E_{BTI}} - \frac{Ea}{k \times T}} \times \\ t\_stress^{0.25} \times (1 - \sqrt{etha \times \frac{t\_rec}{t\_stress + t\_rec}}) \end{aligned} \tag{2}$$

Figure 20 shows the normalized highest dV_{th}_BTI with respect to the theoretical maximum BTI voltage degradation that transistors can suffer after the 3-year lifetime. Such a voltage degradation comes from the NBTI effect (i.e., applying '0' to the gate of PMOS transistors), which has more weight than PBTI (i.e., applying '1' to the gate of NMOS transistors) in the parameter A_{BTI} from Equation 2. Results have been split according to the different types of transistors in the inverter loop of the SRAM cells.

As observed, the BTI degradation is much more noticeable in the PMOS transistors due to the aforementioned reason. Focusing on them, most T_{P1} transistors show a higher dV_{th}_BTI than T_{P2} for a given mechanism. This is due to the stress time in transistors T_{P1} and T_{P2} is given by '0' and '1' cell duty cycles, respectively, and, as analyzed above, the former are normally longer than the latter.

As expected from the '0' duty cycle analysis, the highest BTI degradation in T_{P1} belongs to the baseline approach, followed by BW, SZB, and BW+SZB. In contrast, for transistors T_{P2}, the baseline and SZB obtain very similar results. This is also the case of BW and BW+SZB, which can also be seen in the '1' duty cycle results. Notice too that the degradation on the BW+SZB control bits is also consistent with the previous analysis. Another interesting observation is that the dV_{th}_BTI for the T_{P1} under the baseline scheme (almost) reaches the theoretical maximum degradation in all the studied benchmarks. As mentioned above, this is due to the '0' duty cycle being near 100% in this scheme.

Given the symmetry of the SRAM cell, a similar reasoning as that for the PMOS transistors can also be made for the



a) *PMOS transistors* b) *NMOS transistors*

Fig. 20. Normalized highest dV_{th} caused by the BTI effect on the entire cache for each type of transistor. The cross symbol refers to the normalized highest dV_{th}_BTI in the control bits.

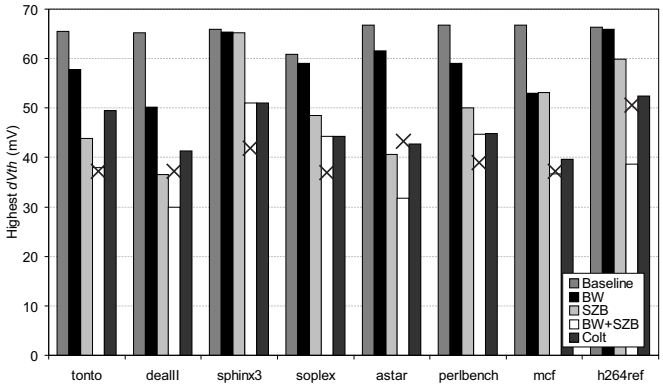


Fig. 21. Highest dV_{th} on the entire cache across the studied applications. The cross symbol refers to the highest dV_{th} in the BW+SZB control bits.

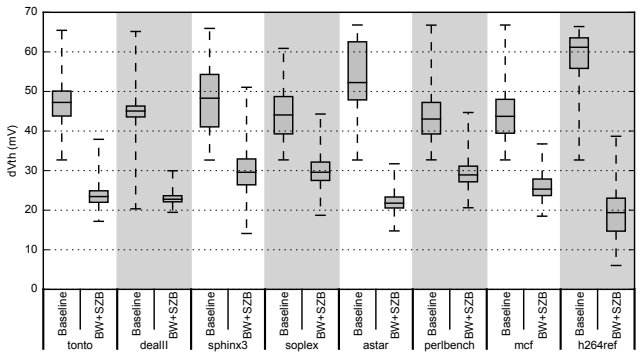


Fig. 22. dV_{th} of all the memory cells for the baseline and BW+SZB mechanisms.

NMOS. In this case, T_{N1} and T_{N2} are affected by the '1' and '0' cell duty cycles, respectively.

3) **Overall dV_{th}:** The overall amount of dV_{th} is calculated as the sum of dV_{th}_HCI and dV_{th}_BTI. Figure 21 depicts the highest raw V_{th} degradation (in mV) across the whole cache for the studied approaches, including the Colt mechanism. As expected, the baseline mechanism is the one with the highest dV_{th}, which ranges from 60 to 70mV. The proposed schemes substantially reduce the dV_{th} with respect to the baseline, especially when combining both BW and SZB jointly. Despite Colt reducing the flip peaks compared to BW+SZB (see Figure 17), its dV_{th}_HCI savings do not compensate for the increase in the dV_{th}_BTI caused by longer duty cycles in applications like *tonto*, *dealII*, and *h264ref*. Moreover, when taking into account the BW+SZB control bits, Colt only saves dV_{th} in the *astar* benchmark. Overall, the proposed BW+SZB guarantees a significant dV_{th} reduction ranging from 21.8% (*h264ref*) to 44.3% (*mcf*) with respect to the baseline approach.

Figure 22 shows the raw dV_{th} results for all the cache cells as a box-and-whisker plot, where the upper and lower bars refer to the maximum and minimum dV_{th} values, respectively. The upper and lower box edges specify the 25th and 75th percentiles of the distribution, whereas the dashed line within the box represents the median of the entire data set.

For all the studied benchmarks, the BW+SZB boxes are located way down below those of the baseline. This shows that the proposed BW+SZB mechanism is effective not only for reducing the dV_{th} in the memory cell with the highest degradation but also for most of the remaining cells implementing the cache (at least for 50% of the cache cells included in the box). On average, BW+SZB saves the dV_{th} from 32.8% (*perlbench*) to 68.3% (*h264ref*).

Notice too that, apart from *h264ref* where the box height for BW+SZB is scarcely larger than that of the baseline, all the benchmarks show a much shorter box height for BW+SZB. This fact denotes a much more homogeneous voltage degradation across half of the cache memory cells. Moreover, taking into account the height of the boxes plus the limits, that is, all the cells used to implement the cache, BW+SZB shrinks such a height in all applications except *sphinx3* compared to the baseline. This homogeneous degradation also ensures a more balanced SNM of the SRAM cells that otherwise can lead to wrong read/write values.

VI. Related Work

Cache degradation has been mainly attacked in the past from the perspective of BTI, by 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 [3], [4], [5], [8], by including redundant cell regions into the cache design [6], and by lowering the supply voltage of idle memory cells [7].

Abella *et al.* [3] invalidate the stored contents and complements their logic values in specific sets and ways. The target sets and ways are selected using a round-robin policy at coarse time periods to mitigate the impact on performance.

Gebregiorgis *et al.* [4] introduce a bit flipping technique that identifies a bit position within the input data with a BTI-optimal signal probability, and uses its logic value as a flag to determine whether to invert the contents of the remaining input bit positions.

Ganapathy *et al.* [5] equalize the duty cycle ratio in read-modify-write cache schemes. Their approach invert the stored contents on writebacks after read operations, whereas regular write accesses do not complement the original stored data.

The *Proactive Recovery* [6] mechanism includes virtual V_{dd} rails in the SRAM cell design to move from normal operation mode to a suspended NBTI wearout mode in which the stored cell logic value depends on which PMOS device is more stressed by the NBTI effect.

Calimera *et al.* [7] partition the cache into multiple banks and uniformly distribute the idle time of the cache lines across them by using a dynamic indexing scheme. The V_{dd} of the idle memory cells is reduced, which mitigates NBTI.

Other approaches have addressed BTI wearout in other microprocessor structures. For example, Wang *et al.* [26] divide the integer register file into two halves and balance the duty cycle distribution in the upper half where most entries hold ‘0’. Gong *et al.* [13] include robust 8T cells in the register file design, which store the most NBTI-vulnerable bits, while the remaining data bits are stored in typical 6T cells. Kothawade *et al.* [27] equalize the NBTI stress across the register file by altering the register decoding scheme. Li *et al.* [28] exploit the idle time of entries from the register file, the reorder buffer, and reservation stations to perform Proactive Recovery [6] during *inactive* cycles. The *Recovery Boosting* [19] technique modifies the SRAM cells used in the issue queue and the register file to enable both PMOS devices within the cell to recover at the same time from NBTI. The GNOMO [29] approach mitigates BTI in the computational units by first completing the execution of a given task faster with a higher voltage/frequency pair, and then powering off such units for the remaining elapsed time.

Other works have taken into account additional parameters that speed up BTI wearout like the temperature [25], [30], the effects of process variation [31], and the proposal of job-to-core mapping techniques [32].

Finally, the Colt approach [8] combats BTI and HCI in L1 caches by periodically inverting the incoming data and uniformly distributing the cache accesses across sets, respectively.

VII. Conclusions

Lifetime reliability is a major design concern in current microprocessors. Two of the main effects that speed up microprocessor wearout are the Hot Carrier Injection (HCI) and the Bias Temperature Instability (BTI). These phenomena progressively degrade the transistor’s threshold voltage, which causes slower transistor switching and eventually can result in faulty operation. HCI is accentuated when transistors flip from logic ‘0’ to ‘1’ and vice versa, whereas BTI exacerbates when transistors maintain the same logic value (i.e., logic duty cycle) for an extended period of time. These failure effects are especially critical in those transistors used to implement the SRAM cells of first-level (L1) caches, which are critical for performance and are continuously aging.

This work has identified which memory cell transistors of the L1 data cache age the most from the perspective of both HCI and BTI effects. Based on this information, this paper has presented a pair of microarchitectural mechanisms that spread the number of cell flips evenly across memory cells and reduce the amount of time that logic ‘0’ values are stored in the cells by switching off specific data bytes.

Experimental results have shown that the proposed mechanisms reduce the highest cell flip of the entire cache from 22.4% to 65.5% depending on the application, whereas the longest ‘0’ and ‘1’ duty cycle distributions in the cells can be reduced up to 40.2% and 20.5%, respectively. This translates into threshold voltage degradation savings ranging from 21.8% to 44.3% depending on the application.

Acknowledgments

This work has been supported by the Spanish *Ministerio de Economía y Competitividad* (MINECO), with Plan E funds under Grant TIN2015-66972-C5-1-R, by a HiPEAC Collaboration Grant funded by the FP7 HiPEAC Network of Excellence under grant agreement 287759, and by the Engineering and Physical Sciences Research Council (EPSRC) through Grants EP/K026399/1 and EP/J016284/1. Additional data related to this publication are available in the data repository at http://dx.doi.org/10.17863/CAM.6183.

References

[1] O. Kononchuck and B. Y. Nguyen, *Silicon-on-Insulator (SOI) Technology: Manufacture and Applications*. Elsevier, 2014.

[2] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, B. Cherkauer, J. Stinson, J. Benoit, R. Varada, J. Leung, R. D. Limaye, and S. Vora, “A 65-nm Dual-Core Multithreaded Xeon Processor With 16-MB L3 Cache,” *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 17–25, 2007.

[3] J. Abella, X. Vera, and A. González, “Penelope: The NBTI-Aware Processor,” in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarch.*, 2007, pp. 85–96.

[4] A. Gebregiorgis, M. Ebrahimi, S. Kiamehr, F. Oboril, S. Hamdioui, and M. B. Tahoori, “Aging Mitigation in Memory Arrays Using Self-controlled Bit-flipping Technique,” in *Proc. 20th Asia South Pacific Design Autom. Conf.*, 2015, pp. 231–236.

[5] S. Ganapathy, R. Canal, A. González, and A. Rubio, “iRMW: A Low-Cost Technique to Reduce NBTI-Dependent Parametric Failures in L1 Data Caches,” in *Proc. 32nd IEEE Int. Conf. Comput. Design*, 2014, pp. 68–74.

[6] J. Shin, V. Zyuban, P. Bose, and T. M. Pinkston, “A Proactive Wearout Recovery Approach for Exploiting Microarchitectural Redundancy to Extend Cache SRAM Lifetime,” in *Proc. 35th Int. Symp. Comput. Arch.*, 2008, pp. 353–362.

[7] A. Calimera, M. Loghi, E. Macii, and M. Poncino, “Dynamic Indexing: Leakage-Aging Co-Optimization for Caches,” *IEEE Trans. Comput.-Aided Design of Integr. Circuits Systems*, vol. 33, no. 2, pp. 251–264, 2014.

[8] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, “Combating Aging with the Colt Duty Cycle Equalizer,” in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarch.*, 2010, pp. 103–114.

[9] R. Ubal, J. Sahuquillo, S. Petit, and P. López, "Applying the Zeros Switch-Off Technique to Reduce Static Energy in Data Caches," in *Proc. 18th Int. Symp. Comput. Arch. High Perform. Comput.*, 2006, pp. 133–140.

[10] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Base-delta-immediate Compression: Practical Data Compression for On-chip Caches," in *Proc. 21st Int. Conf. Paral. Arch. Compil. Tech.*, 2012, pp. 377–388.

[11] N. Khoshavi, R. A. Ashraf, and R. F. DeMara, "Applicability of Power-Gating Strategies for Aging Mitigation of CMOS Logic Paths," in *IEEE 57th Int. Midwest Symp. Circuits and Systems*, 2014, pp. 929–932.

[12] *Standard Performance Evaluation Corporation, available online at http://spec.org/cpu2006*.

[13] N. Gong, S. Jiang, J. Wang, B. Aravamudhan, K. Sekar, and R. Sridhar, "Hybrid-cell register files design for improving NBTI reliability," *Elsevier Microelec. Reliab.*, vol. 52, no. 9–10, pp. 1865–1869, 2012.

[14] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, "CACTI 5.1," *HP Development Company, Palo Alto, CA, USA. Tech. Rep. HPL-2008-20*, 2008.

[15] A. Valero, N. Miralaei, S. Petit, J. Sahuquillo, and T. M. Jones, "Enhancing the L1 Data Cache Design to Mitigate HCI," *To Appear in IEEE Comp. Arch. Letters*, 2015.

[16] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories," in *Proc. Int. Symp. Low Power Electron. Design*, 2000, pp. 90–95.

[17] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," in *Proc. 28th Annu. Int. Symp. Comp. Arch.*, 2001, pp. 240–251.

[18] A. Calimera, E. Macii, and M. Poncino, "Analysis of NBTI-Induced SNM Degradation in Power-Gated SRAM Cells," in *Proc. IEEE Int. Symp. Circuits Systems*, 2010, pp. 785–788.

[19] T. Siddiqua and S. Gurumurthi, "Enhancing NBTI Recovery in SRAM Arrays Through Recovery Boosting," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 4, pp. 616–629, 2012.

[20] M. T. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "ARO-PUF: An Aging-Resistant Ring Oscillator PUF Design," in *Proc. Design, Automat. Test Europe Conf. Exhibit.*, 2014, pp. 1–6.

[21] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design," in *Proc. 43rd ACM/IEEE Design Automat. Conf.*, 2006, pp. 1047–1052.

[22] *Semiconductor Industries Assoc., International Technology Roadmap for Semiconductors, http://www.itrs.net/, 2011*.

[23] R. Ubal, J. Sahuquillo, S. Petit, and P. López, "Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors," in *Proc. 19th Int. Symp. Comput. Arch. High Perform. Comput.*, 2007, pp. 62–68.

[24] E. Mintarno, V. Chandra, D. Pietromonaco, R. Aitken, and R. W. Dutton, "Workload-Dependent NBTI and PBTI Analysis for a sub-45nm Commercial Microprocessor," in *Proc. IEEE Int. Reliab. Physics Symp.*, 2013, pp. 1–6.

[25] A. Tiwari and J. Torrellas, "Facelift: Hiding and Slowing Down Aging in Multicores," in *Proc. 41st Annu. IEEE/ACM Int. Symp. Microarch.*, 2008, pp. 129–140.

[26] S. Wang, T. Jin, C. Zheng, and G. Duan, "Low Power Aging-Aware Register File Design by Duty Cycle Balancing," in *Proc. Design, Automat. Test Europe Conf. Exhibit.*, 2012, pp. 546–549.

[27] S. Kothawade, K. Chakraborty, and S. Roy, "Analysis and Mitigation of NBTI Aging in Register File: An End-To-End Approach," in *Proc. 12th Int. Symp. Quality Electron. Design*, 2011, pp. 1–7.

[28] L. Li, Y. Zhang, J. Yang, and J. Zhao, "Proactive NBTI Mitigation for Busy Functional Units in Out-of-Order Microprocessors," in *Proc. Design, Automat. Test Europe Conf. Exhibit.*, 2010, pp. 411–416.

[29] S. Gupta and S. S. Sapatnekar, "GNOMO: Greater-than-NOMinal Vdd Operation for BTI Mitigation," in *Proc. 17th Asia South Pacific Design Automat. Conf.*, 2012, pp. 271–276.

[30] O. Khan and S. Kundu, "A Self-Adaptive System Architecture to Address Transistor Aging," in *Proc. Design, Automat. Test Europe Conf. Exhibit.*, 2009, pp. 81–86.

[31] X. Fu, T. Li, and J. Fortes, "NBTI Tolerant Microarchitecture Design in the Presence of Process Variation," in *Proc. 41st Annu. IEEE/ACM Int. Symp. Microarch.*, 2008, pp. 399–410.

[32] M. Basoglu, M. Orshansky, and M. Erez, "NBTI-Aware DVFS: A New Approach to Saving Energy and Increasing Processor Lifetime," in *Proc. the Int. Symp. Low Power Electron. Design*, 2010, pp. 253–258.

Alejandro Valero received the BS, MS, and PhD degrees in Computer Engineering from the Universitat Politècnica de València, Spain, in 2009, 2011, and 2013, respectively. He is an Assistant Professor in the Department of Computer and Systems Engineering at the University of Zaragoza, Spain. He was a visiting researcher at Northeastern University, USA, and the University of Cambridge, UK, in 2013-14 and 2014-15, respectively. In 2012, his research was recognized with the Intel Doctoral Student Honor Programme Award. His research interests include memory hierarchy design, energy efficiency, and reliability.

Negar Miralaei received the BS and MS in Computer Hardware Engineering and Computer Architecture Engineering, respectively, from Shahid Rajayee Univeristy, Tehran, and Shahid Beheshti University, Tehran, Iran, in 2008 and 2012. She was a 4th year PhD Student in Computer Science at the University of Cambridge, UK. Her research interests included reliability in computer architecture, emerging challenges in new processes technologies, and VLSI CAD design.

Salvador Petit received the PhD degree in Computer Engineering from the Universitat Politècnica de València (UPV), Spain. Since 2009, he is an Associate Professor in the Computer Engineering Department at the UPV, where he has taught several courses on computer organization. He has published more than 100 refereed conference and journal papers. In 2013, he received the Intel Early Career Faculty Honor Program Award. His research topics include multithreaded and multicore processors, memory hierarchy design, task scheduling, as well as real-time systems. Prof. Petit is a member of the IEEE Computer Society.

Julio Sahuquillo received the BS, MS, and PhD degrees in Computer Engineering from the Universitat Politècnica de València, Spain. Since 2002 he is an Associate Professor in the Department of Computer Engineering at the same university. He has taught several courses on computer organization and architecture. He has published more than 120 refereed conference and journal papers. His current research topics include multi- and manycore processors, memory hierarchy design, cache coherence, GPU architecture, and architecture-aware scheduling. He is a member of the IEEE Computer Society.

Timothy M. Jones received an MEng degree in Computer Science from the University of Bristol and a PhD in Informatics from the University of Edinburgh. He currently holds an EPSRC Early Career Fellowship at the University of Cambridge Computer Laboratory. Prior to this he held a Postdoctoral Research Fellowship from the Royal Academy of Engineering and EPSRC, and has been a visiting researcher at Harvard University. His research interests include computer architecture and compiler optimization, with a focus on extracting all forms of parallelism, energy efficiency, and reliability. He is a member of the IEEE Computer Society.