NBTI Recovery in SRAM Arrays through Enhanced Recovery Boosting

Mudasar Basha
Assistant Professor E.C.E Department BVRIT, Narsapur, Hyderabad502313, A.P,India

Shaik Karimullah
Assistant Professor, ECE Dept., AITS, Rajampet, Kadapa, A.P, India

K.Abdul Rehman
Assistant Professor E.C.E Department Dr.KVSRIT, Opp Dupaadu Railway station Kurnool- A.P, India

Abstract—Negative bias temperature instability (NBTI) is an important lifetime reliability problem in microprocessors. SRAM-based structures within the processor are especially susceptible to NBTI since one of the pMOS devices in the memory cell always has an input of "0". Previously proposed recovery techniques for SRAM cells aim to balance the degradation of the two pMOS de-vices by attempting to keep their inputs at a logic "0" exactly 50% of the time. However, one of the devices is always in the negative bias condition at any given time. In this paper, we propose a technique called Recovery Boosting that allows both pMOS devices in the memory cell to be put into the recovery mode by slightly modifying to the design of conventional SRAM cells. We evaluate the circuit-level design of a physical register file and an issue queue that use such cells through SPICE-level simulations. We then conduct an architecture-level evaluation of the performance and reliability of using area-neutral designs of these two structures. We show that Recovery Boosting provides significant improvement in the static noise margins of the register file and issue queue while having very little impact on power consumption and performance.

Index Terms—Negative bias temperature instability (NBTI), static random access memory (SRAM).

I. INTRODUCTION

Reliability is one of the biggest challenges facing the microprocessor industry today. With continued technology scaling, processors are becoming increasingly susceptible to hard errors. Hard errors are permanent faults that occur due to the wearing out of hardware structures over time. These failures occur partly due to design-time factors such as process parameters and wafer packaging, as well as runtime factors such as the utilization of the hardware resources and the operating temperature. It is important to ensure that the re-liability of the micro-architectural structures in the processor is maximized so that one can make use all the available hardware resources effectively over the entire service life of the chip.

One important hard error phenomenon is negative bias tem-perature instability (NBTI), which affects the lifetime of pMOS transistors. NBTI occurs when a negative bias (i.e., a logic input of "0") is applied at the gate of a pMOS transistor. The negative bias can lead to the generation of interface traps at the Si/SiO_2 interface, which cause an increase in the threshold voltage of the device. This increase in the threshold voltage degrades the speed of the device and reduces the noise margin of the circuit, eventually causing the circuit to fail [11], [14]. One interesting aspect of NBTI is that some of the interface traps can be eliminated by applying a logic input of "1" at the gate of the pMOS device. This puts the device into what is known as the recovery mode, which has a "self-healing" effect on the device [1].

Memory arrays that use static random access memory (SRAM) cells are especially susceptible to NBTI. SRAM cells consist of cross-coupled inverters that contain pMOS devices. Since each memory cell stores either a "0" or a "1" at all times, one of the pMOS devices in each cell always has a logic input of "0." Since modern processor cores are composed of several critical SRAM-based structures, such as the register file and the issue queue, it is important to mitigate the impact of NBTI on these structures to maximize their lifetimes. Previous work on applying recovery techniques to SRAM structures aim to balance the degradation of the two pMOS devices in a memory cell by attempting to keep the inputs to each device at a logic input of "0" exactly 50% of the time [1], [11], [19]. However, one of the devices is always in the negative bias condition at any given time. In this paper, we propose a novel technique called Recovery Boosting that allows both pMOS devices in the memory cell to be put into the recovery mode. The remainder of this paper is organized as follows.. Section II discusses the related work on NBTI mitigation techniques. Section III discusses the recovery boosting technique and the circuit-level design, and evaluation of the register file and issue queue are given in Section IV. The

experimental methodology used for the architecture-level evaluation is given in Section V, and the corresponding results in Section VII. Section VIII concludes this paper.

II. RELATED WORK

There are two basic approaches to mitigating NBTI: 1) reduce the stress on the pMOS transistors and 2) enhance the recovery process. Stress reduction techniques aim to reduce the aging rate by controlling V_{dd}, V_t, and temperature, whereas recovery enhancement techniques aim to increase the recovery time for the pMOS devices. One could implement these techniques at a coarse granularity (e.g., for entire cores) [7], [21], [22] or for in-dividual structures within the core [1], [9], [11], [19]. Recovery boosting is a recovery-enhancement technique for SRAM struc-tures.

1) Stress Reduction Techniques: Srinivasan et al. [21] esti-mate the MTTF due to aging mechanisms at runtime based on the operating conditions and use dynamic reliability manage-ment (DRM) to stay within the reliability budget. Tiwari and Torrellas propose a technique called "Facelift" [22] to hide the effects of aging through temperature-based job-scheduling to in-dividual cores of a multicore processor, in conjunction with V_{dd} and V_t control. Feng et al. [7] propose to use on-chip reliability sensors to guide job-scheduling decisions on a multicore pro-cessor. The design of such on-chip NBTI sensors are discussed in [5]. The use of stress reduction techniques is orthogonal to the use of recovery enhancement.

2) *Recovery-Enhancement Techniques:* Abella et al. [1] pro-pose to feed specific bit patterns into the devices to increase the recovery time for pMOS transistors in logic structures (e.g., adders) during idle periods and balance the degradation of the pMOS devices in SRAM-based memory structures when they hold invalid data. Kumar et al. [11] propose a similar technique to periodically flip the contents of SRAM cells to balance the wear on the pMOS transistors. Shin et al. [19] propose a re-covery enhancement technique for caches where SRAM cells are proactively put into the recovery mode via the use of a spare
memory array. They reverse bias $(V_s = V_{dd})$ the pMOS de-vices to put them into a deep recovery state. When an array is put into the recovery mode, the pMOS devices in one of the in-verters in all of the cells are put into the recovery mode followed by those in the other inverter and this recurring pattern is con-tinued throughout the recovery period for the array. All of these recovery enhancement techniques aim to balance the degrada-tion of the two pMOS devices in the memory cell by attempting to keep the inputs to each device at a logic input of "0" (i.e., negative-bias) exactly 50% of the time.

A recently issued U.S. patent describes an idea similar to one of our proposed recovery-boosting technique (coarse-grained) which is discussed in Section IV [4]. However, the idea in the patent has the following drawback: it takes multiple processor clock cycles to put both PMOSs into recovery and therefore cannot be used for high-speed SRAM arrays. Moreover, the patent does not provide any analysis of the impact.

III. BASICS OF RECOVERY BOOSTING

Before we discuss recovery boosting, we first review the design and operation of a conventional 6-transistor (6T) SRAM cell. The design of the 6T cell is given in Fig. 1(a). The cell is composed of a wordline (WL), a pair of bitlines (BL, BLB), two cross-coupled inverters (N_0, P_1), and two access transistors (N_0, N_1). The cross-coupled inverters store one bit of data. There are three basic operations that one can perform on this SRAM cell: read, write and hold. To read and write data, the cell is selected by raising WL to high. This activates the access transistors and connects the inverters in the cell to the bitlines. During a read operation, both bitlines are first precharged high. Based on the data stored in the cell, one of the bitlines is discharged. This change is detected by a sense amplifier (which is not part of the cell) to determine the value stored in the cell. During a write operation, one of the bitlines is raised high and the other is lowered depending on the value to be written to the cell. When the cell is not selected $(WL = 0)$ for read or write, it is expected to hold the data stored in it and is said to operate in the hold mode.

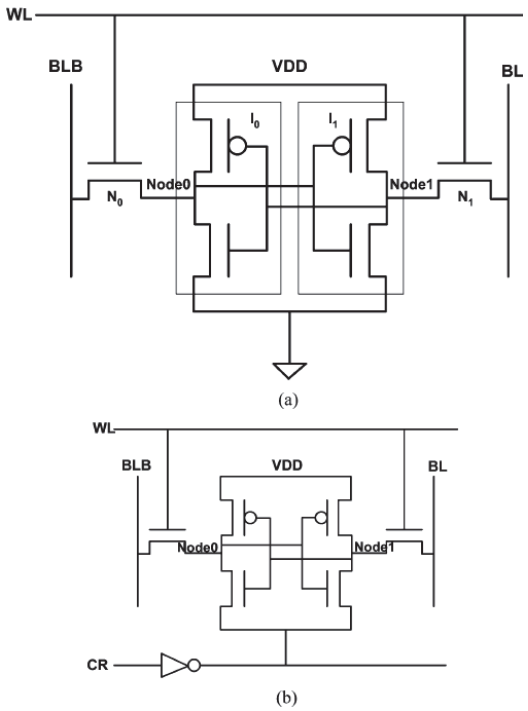


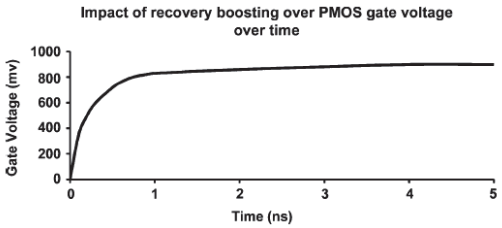
Fig. 1. SRAM cell design for recovery boosting. (a) Conventional 6T SRAM cell. (b) Modified SRAM cell that supports recovery boosting.

TABLE I
MODIFIED SRAM CELL OPERATION

CR	WL	BL	BLB	Node0	Node1	Operation
1	0	X	X	0/1	1/0	Hold
1	1	1	1	0/1	1/0	Read
1	1	1	0	0	1	Write '1'
1	1	0	1	1	0	Write '0'
0	X	X	X	1	1	Recovery Boost

However, the drawback of this approach is that it can take a long time to raise both the node voltages to V_{dd} in a high-perfor-mance processor that operates at a high clock frequency. This is illustrated in Fig. 2, which presents the achieved pMOS gate voltages of a bitcell over time due to recovery boosting. The simulation is performed using the Cadence Virtuoso Spectre cir-cuit simulator for the 32-nm process using the Predictive Tech-nology Model.3 The operating temperature is 90 $_{o}$ C, which is the average temperature in which the high-performance proces-sors operate [12]. We use this temperature value throughout the paper for all the experiments. We can observe that this approach achieves the desired gate voltage (V_{dd}) within 3.33 ns. For a pro-cessor which operates at 3-GHz frequency, it will take ten cycles to switch to the recovery boost mode. Similarly, it takes around ten cycles to go back to the normal operating mode from the re-covery boost mode. However, our goal is to be able to switch be-tween the recovery boost mode and the normal operating mode within a single cycle which is critical for a high-speed SRAM structure, such as the issue queue, where instructions need to be woken up and selected within a single clock cycle, in order to expedite the execution of dependent instructions. As men-tioned before, recovery boost mode is applied when an entry of the structure holds data that is considered "invalid" at the archi-tecture-level. Entries in the high-speed structures change their status between valid and invalid very frequently. For example, we find from architecture simulations that an issue queue entry stays invalid for about 50 cycles before it changes its status to valid. In such scenario, the cell shown in Fig. 1(b) will take 20 cycles of the 50 cycles (40% of the invalid period) to shift be-tween modes, given that shifting to the normal operating mode takes place during the end of the invalid period. Thus, only 30 cycles could be utilized for the recovery process. On the other hand, if extra cycles are allocated to shift to the normal operating mode after the invalid period, that would have neg-ative consequences on the processor performance. Therefore, single-cycle

switching is required for the high-speed structures in the processor for the maximum utilization of the invalid states for the recovery process without any performance loss. Such single-cycle switching can be achieved by raising the bitlines along with the ground voltage to V_{dd}. There are various ways of incorporating such cells into SRAM arrays, which we will dis-cuss shortly.



IV. EXPERIMENTAL METHODOLOGY FOR THE ARCHITECTURE-LEVEL ANALYSIS

Having seen the circuit-level design of the physical register file and the issue queue to support recovery boosting, we now evaluate the impact of using these techniques at the architecture level. We carry out our architecture-level evaluations via execu-tion-driven simulation using the M5 simulator [3]. We use the system-call emulation mode of M5. Our workloads consist of all 26 benchmarks from the SPEC CPU2000 benchmark suite

. The benchmarks are compiled for the Alpha ISA and use the reference input set. We perform detailed simulation of the first 100-million instruction SimPoint for each benchmark [18]. We model a four-wide issue core, which is similar to those in mul-ticore processors. We assume the initial threshold voltage of the pMOS devices in the memory cells to be 0.2 V and the service life of the processor to be seven years based on the work by Ti-wari and Torrellas [22].

1) Reliability Metric—Read Static Noise Margin (SNM):

NBTI causes an increase in the threshold voltage of the pMOS transistors. In the case of SRAM cells, this shift in V_t could increase the time needed for reading from and writing to the cells. NBTI could also decrease the read SNM of the cells. The SNM is a measure of the stability of the cell and specifies the maximum amount of voltage noise that can be tolerated at the nodes of the memory cell before the contents of the cell get flipped [16]. Previous work [11] has shown that, of these three metrics, the SNM is the one that is most heavily affected by NBTI and therefore we use SNM as the reliability metric in this paper.

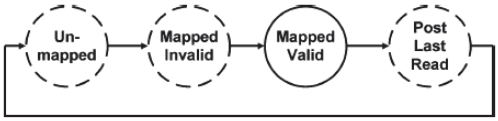
Initially, before the processor is used for executing work-loads, the bitcells in the register file and the issue queue are designed such that their SNM is not limited by the strength of the pMOS devices. But after these structures are exercised by workloads, their SNM gets limited by the strength of the pMOS devices due the impact of NBTI on V_t. We capture this impact by tracking the stress and recovery cycles on all of the pMOS devices in the register file and the issue queue (based on our cir-cuit-level designs of these structures) over the course of an ar-chitecture simulation and extrapolate the statistics to calculate the degradation in V_t after the seven-year service life. We then feed the V_t values of these pMOS devices into the Cadence Vir-tuoso Spectre circuit simulator to calculate the SNM of all the cells in a structure at the end of the seven-year period and use the smallest value to denote the SNM for that structure.

V. ARCHITECTURE-LEVEL SIMULATION RESULTS

We now study the impact of putting memory cells of regis-ters and issue queue entries into the recovery boost mode when they hold invalid data. As discussed earlier, we put registers into the recovery boost mode when they are in theUnmapped and Mapped-Invalid states. We present results only for the in-teger register file for two reasons. First, the integer benchmarks have very few floating-point instructions and therefore rarely exercise the floating-point register file. Second, several of the floating-point benchmarks have a large number of integer in-structions in their first 100-million instruction SimPoint and therefore significantly exercise the integer register file. (Nine out of the 14 floating-point workloads have more integer than floating-point instructions in their first SimPoint.) As a result, the integer register file is exercised to a greater extent than the the floating-point register file for most of the benchmarks. Issue queue entries are put into the recovery boost mode when they no longer hold valid data.

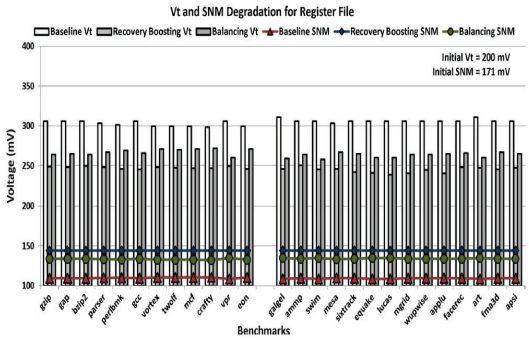
We evaluate three different processor configurations, which we denote as:Baseline, Recovery Boosting and Balancing. Base-line models four-wide issue core that do not use any NBTI mit-igation technique. Recovery Boosting replaces the integer reg-ister file and the issue queue of the baseline configuration with their counterparts that support recovery boosting. We assume that the modified structures for Recovery Boosting are designed to be area-neutral with respect to Baseline by trading off a small amount of storage capacity to accommodate the extra area re-quired to implement recovery boosting. Based on our area evalu-ations in Section V, we assume that the modified integer register file and issue queue have 125 registers and 62 entries, respec-tively.

For the remainder of this paper, we will refer to the integer register file and the issue queue as RF and IQ, respectively.

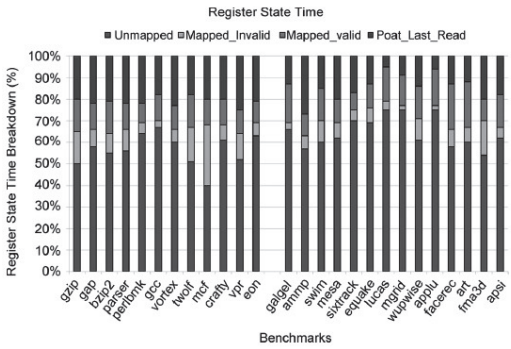


A. Physical Register File Results

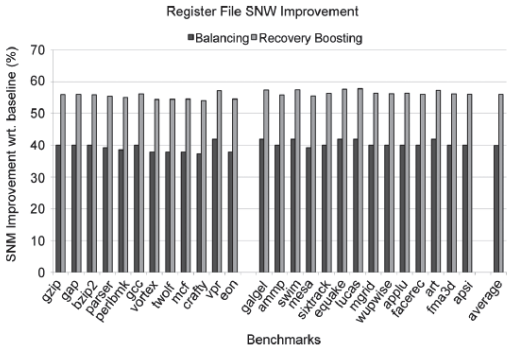
Fig. 13 shows the V_t degradation and the resultant SNM for the RF across the benchmarks due to NBTI after the seven-year service life for the Baseline, Recovery Boosting, and Balancing configurations. As mentioned earlier, the initial V_t is 200 mV and, using this value, we get the initial SNM of the RF to be 171 mV before the RF is stressed by the benchmark. For the Baseline, on an average, the V_t degrades to 305 mV across the benchmarks, which leads to an average SNM of 109 mV. There-fore, Baseline causes about 37% degradation in SNM.



To im-prove the SNM over the Baseline, Recovery Boosting takes ad-vantage of the invalid periods of the RF to apply recovery to the registers. A breakdown of the time spent by the registers in the four different states given in Fig. 5 is shown in Fig. 14. The values given in the graph are an average over all the reg-isters and over the entire SimPoint for each benchmark. As we can see, the registers are in the Unmapped and Mapped-Invalid states for a large fraction of time for most of the benchmarks. Therefore, we have significant opportunities to apply recovery boosting for the RF. The impact on the SNM as a result of using recovery boosting is given in Fig. 13. As Fig. 13 shows, the av-erage degraded V_t stays close to 245 mV because of the applied recovery to the RF bitcells. This causes the SNM to degrade to an average value of 144 mV, which is about 15% degradation in SNM over the initial condition. Similarly, for the Balancing approach, the average degraded V_t stays close to 265 mV and the SNM degrades to an average value of 133 mV.



In Fig. 15, each pair of bars shows the improvement in the SNM over Baseline for Recovery Boosting and Balancing, respectively. As Fig. 15 shows, while Balancing provides a good improvement in the SNM, Recovery Boosting provides significantly higher reliability benefits by virtue of its ability to



VI. CONCLUSION AND FUTURE WORK

NBTI is one of the most important silicon reliability problems facing processor designers. SRAM memory cells are especially vulnerable to NBTI since the input to one of the pMOS devices in the cell is always at a logic "0." In this paper, we propose re-covery boosting, a technique that allows both pMOS devices in the cell to be put into the recovery mode by raising the ground voltage and the bitline to V_{dd}. We show how fine-grained re-covery boosting can be used to design the physical register file and issue queue and evaluate their designs via SPICE-level sim-ulations. We then show that area-neutral designs of these two structures can provide significant reliability benefits with very little impact on power consumption and negligible loss in per-formance.

The fine-grained recovery boosting approach that we eval-uate in this paper can be used for small SRAM arrays. In future work, we plan to study the use of coarse-grained recovery boosting, which imposes less area overheads, for designing caches. Caches pose additional challenges, such as identi-fying when lines become valid to put them into the recovery boost mode. We plan to explore the use of techniques such as dead-block prediction [10] in conjunction with recovery boosting to mitigate the impact of NBTI on caches. We also plan to study the circuit-level design of recovery boosting in depth.

REFERENCES

[1] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-aware pro-cessor," in *Proc. 40th IEEE/ACM Int. Symp. Microarchitecture*, 2007.
[2] H. Akkary, R. Rajwar, and S. T. Srinivasan, "Checkpoint processing and recovery: Towards scalable large instruction window processors," in *Proc. Int. Symp. Microarchitecture (MICRO)*, Dec. 2003, pp. 423–434.
[3] N. L. Binkert *et al.*, "The M5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul. 2006.
[4] P. Bose, J. Shin, and V. Zyuban, "Method for Extending Lifetime Re-liability of Digital Logic Devices Through Removal of Aging Mecha-nisms," U.S. Patent 7 489 161, Feb. 10, 2009.
[5] A. Cabe, Z. Qi, S. Wooters, T. Blalock, and M. Stan, "Small embed-dable NBTI sensors (SENS) for tracking on-chip performance decay," in *Proc. Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2009, pp. 1–6.
[6] O. Ergin, D. Balkan, D. Ponomarev, and K. Ghose, "Increasing pro-cessor performance through early register release," in *Proc. Int. Conf. Comput. Design (ICCD)*, Oct. 2004, pp. 480–487.
[7] S. Feng, S. Gupta, and S. Mahlke, "Olay: Combat the signs of aging with introspective reliability management," in *Proc. Workshop Quality-Aware Design (W-QUAD)*, 2008.
[8] D. Folegnani and A. Gonzalez, "Energy-effective issue logic," in *Proc. Int. Symp. Comput. Architecture (ISCA)*, Jun. 2001, pp. 230–239.

[9] X. Fu, T. Li, and J. Fortes, "NBTI tolerant microarchitecture design in the presence of process variation," in *Proc. Int. Symp. Microarchitec-ture (MICRO)*, Nov. 2008, pp. 399–410.
[10] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting gen-erational behavior to reduce cache leakage power," in *Proc. Int. Symp. Comput. Architecture (ISCA)*, Jun. 2001, pp. 240–251.
[11] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of NBTI on SRAM read stability and design for reliability," in *Proc. Int. Symp. Quality Electron. Design*, 2006, pp. 210–218.
[12] Y. Li, D. Brooks, Z. hu, and K. Skadron, "Performance, energy and thermal considerations for SMT and CMP architectures," in *Proc. Int. Symp. High-Performance Comput. Architecture (HPCA)*, 2005, pp. 71–82.
[13] S. Palacharla, "Complexity-effective superscalar processors," Ph.D. dissertation, Dept. Comput. Sci., Univ. of Wisconsin, Madison, 1998.
[14] S. Park, K. Kang, and K. Roy, "Reliability implications of bias-tem-perature instability in digital ICs," *IEEE Design Test of Comput.*, pp. 8–17, Dec. 2009.
[15] G. Reimbold *et al.*, "Initial and PBTI-induced traps and charges in Hf-based oxides/TiN stacks," *Microelectron. Reliabil.*, vol. 47, no. 4–5, pp. 489–496, Apr. 2007.
[16] E. Seevinck, F. J. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, Oct. 1987.
[17] J. P. Shen and M. H. Lipasti, *Modern Processor Design: Fundamentals of Superscalar Processors (Beta Edition)*. New York: McGraw Hill, 2003.
[18] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for life-time reliability-aware microprocessors," in *Proc. Int. Symp. Comput. Architecture (ISCA)*, Jun. 2004, pp. 276–287.
[19] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *Proc. Int. Symp. Microarchitecture (MICRO)*, Nov. 2008, pp. 129–140.
[20] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Trans. Device Mater. Reliabil.*, vol. 7, no. 4, pp. 509–517, 2007.