Recovery Boosting: A Technique to Enhance NBTI Recovery in SRAM Arrays

Taniya Siddiqua and Sudhanva Gurumurthi
Department of Computer Science
University of Virginia
Email: {taniya,gurumurthi}@cs.virginia.edu

***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 devices 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 the design of conventional SRAM cells. We present the circuit-level design of an issue queue that uses such cells and perform SPICE-level simulations to verify its functionality and quantify area and power consumption. We then conduct an architecture-level evaluation of the performance and reliability of using an area-neutral design of such an issue queue using the M5 simulator and the SPEC CPU2000 benchmark suite. We show that recovery boosting provides a 56% improvement in the static noise margin for the issue queue while having very little impact on power consumption and a negligible loss in performance.**

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. One important hard error phenomenon is Negative Bias Temperature 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 [10], [6]. 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. 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], [6], [12]. 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 main contributions of this paper are:

- We describe how SRAM cells can be modified to support recovery boosting and discuss several circuit and microarchitecture level design considerations when using such cells to build SRAM arrays.
- We present the circuit-level design of an issue queue for a 4-wide issue processor core that uses the modified cells to provide recovery boosting. We verify the functionality of this design and quantify its area and power consumption through SPICE-level simulation using the Cadence Virtuoso Spectre circuit simulator [4] for the 32nm process technology. We show that the modified SRAM structure imposes only a 3% area overhead over the baseline non-recovery boost design and that its maximum power consumption is less than 2% over the baseline.
- We then evaluate the performance and reliability of an area-neutral design of the issue queue at the architecture-level via execution-driven simulation using the M5 simulator [2] and the SPEC CPU2000 benchmark suite [14]. We show that recovery boosting provides a 56% improvement in the static noise margin for the issue queue suite while having a negligible impact on performance.

The organization of the rest of this paper is as follows. The next section discusses the recovery boosting technique. The circuit-level design and evaluation of the issue queue is given in Section III. The architectural experimental methodology is given in Section V and the corresponding results in Section VI. Section VII discusses related work and Section VIII concludes this paper.

II. Basics of Recovery Boosting

Since the SRAM cell has cross-coupled inverters, each inverter charges the gate of the PMOS or NMOS device of the other inverter. Therefore, at any given time, one PMOS device will always be in the stress mode. The goal of recovery enhancement is to put the PMOS devices into the recovery mode by feeding input values to the cell that will transition them into that mode. However, due to the cross-coupled nature of the inverters, only one of the PMOS devices can be put into the recovery mode. We propose a 6T SRAM cell design shown in Figure 1 which is capable of normal operations (read, write, and hold) as well as providing an NBTI recovery mode that we call the *recovery boost mode*

where both PMOS devices within the cell undergo recovery at the same time.

The basic idea behind recovery boosting is to raise the node voltages (Node0 and Node1 in Figure 1) of a memory cell in order to put both PMOS devices into the recovery mode. This can be achieved by raising the ground voltage and bitlines to the nominal voltage through an external control signal. Raising the bitlines to V_{dd} allows for a fast transition into the recovery boost mode, which is important for high-speed SRAM. The modified SRAM cell has the ground connected to the output of an inverter, as shown in Figure 1. CR is the control signal to switch between the recovery boost mode and the normal operating mode. During the normal operating mode, CR has a value of '1' (V_{dd}), which in turn connects the ground of the SRAM cell to a value of '0'. With this connection, the SRAM cell can perform normal read, write, and hold operations. To apply recovery boosting, CR has to be changed to a '0' in order to raise the ground voltage of the SRAM cell to V_{dd}. To raise the voltages of both Node0 and Node1 to V_{dd}, BL and BLB have to be charged to V_{dd} along with the raised ground voltage. This circuit configuration puts both PMOS devices in the SRAM cell into the recovery mode. A cell can be put into the recovery boost mode regardless of whether its wordline (WL) is high or low. Unlike read and write operations on a cell, putting a cell into the recovery boost mode does not require an access to its wordline.

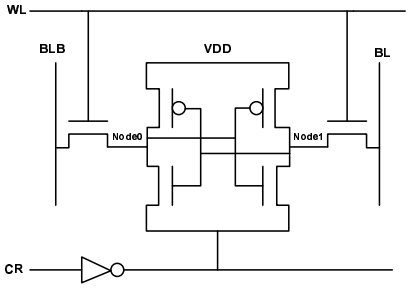


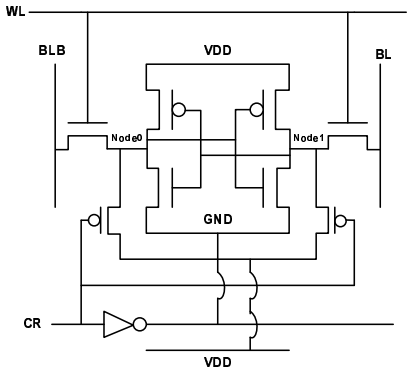
Fig. 1. SRAM Cell Design for Recovery Boosting

Incorporating the modified SRAM cell into a memory array requires additional microarchitectural design considerations. Recovery boosting can be provided at a fine granularity, such as for individual entries/rows of a memory array, or at a coarser granularity, such as for an entire array.

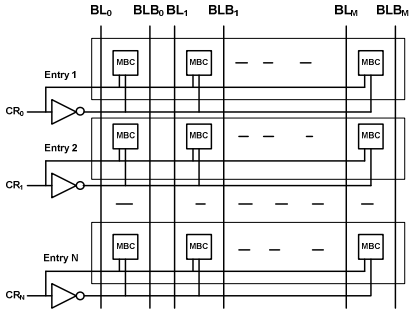
• ***Fine-Grained Recovery Boosting:*** In the normal operating mode, the state of the bitlines change during read and write operations. Since a pair of bitlines is shared by all the memory cells in a given column in the array, even those memory cells that are not being read from or written to will have the voltage on their bitlines changing. In an ordinary SRAM array, these bitline transitions do not affect the normal operation of the cells. However, in order to perform recovery boosting of a memory cell, both bitlines of the cell need to be raised to V_{dd}. Therefore, we need to be able to isolate the bitlines of the memory cells that are in the recovery boost mode from the bitlines that are used for accessing other cells in the array. To provide this isolation, we extend the memory cell in Figure 1 with connections to the V_{dd} rail of an adjoining row or column via two PMOS access devices. The design of the modified SRAM cell is shown in Figure 2(a) and an SRAM array that uses this cell for controlling individual entries to operate either in normal or recovery boost mode is shown in Figure 2(b).

In the memory cell design given in Figure 2(a), the CR signal serves the same purpose as before. When a value of '0' is input to the CR line to transition the cell into the recovery boost mode, in addition to raising the ground voltage, the two extra PMOS devices connected to the V_{dd} rail are also turned on. Therefore, by raising the ground and connecting the bitcell to V_{dd}, the cell can be transitioned into the recovery boost mode without affecting cells in other rows of the array.

NOTE: We make the extra PMOS devices resilient against NBTI by using high-V_t transistors. Although high-V_t devices are slower, these devices are used only when transitioning the cell into the recovery boost mode and not when transitioning to the normal operating mode. Therefore, these devices do not impact performance but may delay the transition into the recovery boost mode. Moreover, since these devices do not lie on the performance critical path, they are sized so as to minimize the overall area. However, the PMOS devices do consume leakage power. We quantify the power consumption in Section III.



(a) Modified SRAM cell with connection to the V_{dd} rail of an adjoining row



(b) SRAM array with modified cell (N entries, M-wide)

Fig. 2. SRAM Array for Fine-Grained Recovery Boosting

• ***Alternative Cell Designs:*** In addition to the cell in Figure 2(a), we considered two alternative SRAM cell designs to support fine-grained recovery boosting, which differ in terms of their area, power, and the time needed to put the cell into the recovery boost mode.

Alt-Cell-1: In this design, only GND is raised to V_{dd}. This design will save area and power as the additional PMOS access devices will no longer be needed. However, the drawback of this approach is that it takes longer (i.e., multiple processor clock cycles) to raise both the node voltages to V_{dd}. Our goal is to be able to switch between the recovery boost mode and the normal operating modes within a single cycle, which is critical for a structure such as the issue queue where instruction wakeup and select need to occur within a single clock cycle for performance reasons.

Alt-Cell-2: This cell design involves a sequence of operations

as follows. The bitlines are first precharged and then a pulse is applied to the respective wordline, which would help both the nodes reach V_{dd}. Finally, the GND is changed to V_{dd} so that the node voltages are held high. Again, this aproach would be efficient in terms of area and power, but there would be write-port contention between normal accesses to the SRAM array and accesses related to recovery actions, which can degrade performance. *Alt-Cell-2* also increases the complexity of the peripheral circuitry.

In this paper, we use the SRAM cell shown in Figure 2(a) since it is better suited for high-speed SRAM arrays.

• ***Coarse-Grained Recovery Boosting:*** In this approach, we use the SRAM cell design shown in Figure 1 instead of the one for fine-grained control. Here, a single control signal puts the entire array into the recovery boost mode. The control signal CR with a value of '0' raises the ground connection of each entry to V_{dd}. In this design, connections to the V_{dd} rail via the PMOS devices are not required. Instead we merely need to raise all the bitlines in the array to V_{dd} to transition all the cells in the array to the recovery boost mode.

• ***Tradeoffs Between Fine-Grained and Coarse-Grained Recovery Boosting:*** Going in for the fine-grained approach entails an area overhead of having two additional PMOS devices for each memory cell which can be prohibitive for large SRAM arrays such as caches. On the other hand, the fine-grained approach provides greater flexibility in managing NBTI by exploiting the usage characteristics of individual entries in the structure. In this paper, we evaluate the use of recovery boosting for the issue queue. Due to the relatively small size of this structure (compared to caches), we use the fine-grained approach.

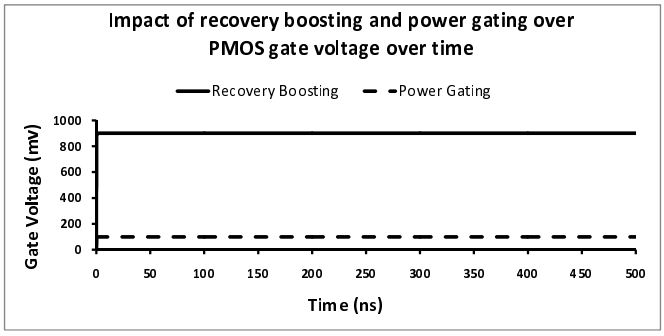


Fig. 3. PMOS gate voltages of an SRAM bitcell due to recovery boosting and power gating (V_{dd}=0.9V)

• ***Difference Between Recovery Boosting and Power Gating:*** Similar to recovery boosting, power gating also involves a small change to the design of the SRAM cell and can also be used to combat NBTI [16]. Power gating helps reduce the stress on the PMOS devices of the SRAM bitcell. However, recovery boosting allows for recovery, since the gate voltage of the PMOS device reaches V_{dd}. This is illustrated in Figure 3 which presents the achieved PMOS gate voltages of a bitcell over time due to recovery boosting and power gating. The simulation is performed using the Cadence Virtuoso Spectre circuit simulator [4] for the 32nm process using the Predictive Technology Model [8]. We can observe that recovery boosting achieves the desired gate voltage (V_{dd}) within a very short interval of time (195 ps), whereas power gating achieves only about 11% of V_{dd} even after several hundred nanoseconds. When a memory cell stores valid data, neither recovery boosting nor power gating can be applied and the PMOS devices in the cells will be stressed in a similar way. However, as Figure 3 shows, when the memory cell is idle and the data in the cell is no longer needed, it would be more beneficial to take advantage of recovery boosting.

• ***Impact of Process Variation on Correct Functionality:*** In deep submicron technologies, intra-die process variation is an important issue. Different transistor parameters, including V_t, are affected by process variation and can impact circuit delay characteristics. V_t is affected by variations in the device geometry, random dopant number flucations, and mobile charges in the gate oxide [13]. Process variation will affect the delay characteristics of the 6T SRAM cell inherent in both the original and modified bitcells in a similar way. Process variation can also impact the V_t of the two V_{dd}-rail access transistors and the devices in the inverter connected to the CR line. If the V_t of the V_{dd} rail transistors is high, then transitioning the bitcell to the recovery boost mode may be slower. This could reduce the amount of time for which we can apply recovery boosting but will not affect the correctness of the SRAM cell operation. On the other hand, if their V_t is lower, the bitcell will transition into the recovery boost mode faster and the access devices will consume higher leakage power but will again not affect correctness. A delay in the PMOS device of the CR line inverter will again merely slow the transition into the recovery boost mode. However, a delay in the NMOS device in the inverter could affect the speed at which the cell transitions out of the recovery boost mode and into a normal operating mode, which can affect correctness. In order to handle this situation, we need to set the clock frequency such that this delay can be accomodated within a single cycle.

• ***Recovery Boosting Does Not Exacerbate PBTI:*** Putting the PMOS devices into the recovery mode does not increase the stress on the NMOS devices in the memory cell. Stresses on the NMOS devices can lead to a phenomenon that is similar to NBTI called PBTI (Positive Bias Temperature Instability), which occurs when a positive bias (V_{gs} = V_{dd}) is applied to the NMOS device. As with NBTI, PBTI also generates interface traps and increases the threshold voltage. PBTI is expected to become more important in future deep submicron technologies [9]. While in the recovery boost mode, both the ground and node voltages of the cell are raised to V_{dd}. Consequently, the V_{gs} of the NMOS devices in the inverters becomes zero and therefore these NMOS devices do not experience any positive bias. The access transistors are also not accessed during the recovery boost mode. Therefore recovery boosting does not exacerbate PBTI on the NMOS devices in the memory cell (and may in fact provide PBTI recovery [9]).

III. Issue Queue Design that Supports Recovery Boosting

Recovery boosting can be implemented for any SRAM structure and is especially well-suited for multiported structures. Although the modifed bitcell has extra devices, which increases the area overheads, the impact of these overheads is less for multiported SRAM structures since only one pair of V_{dd} rail access devices are required per bitcell. Since most large high-speed SRAM arrays within the processor core are heavily multiported to support instruction/thread-level parallelism, the area overheads due to recovery boosting are small. In this paper we present the design and evaluation of one such high-speed multiported SRAM structure: the issue queue.

In this section, we present the design of an issue queue for a 4-wide issue processor core and evaluate this design at the circuit-level. Our baseline design, which we extend to support

recovery boosting, is based on complexity-effective designs (with respect to energy and delay) that have been published previously [7], [5]. The issue queue uses a non-data-captured design and consists of 64 entries with 4 read-ports, 4 write-ports, and 65 bits per-entry. The choice for the entry-size is based on the issue queue descriptions given in [1].

The issue queue houses instructions that have been fetched, decoded, and renamed and are pending execution. Instructions are dynamically scheduled from the issue queue based on the availability of their source operands and functional units. Instructions whose dependencies have been satisfied need to be woken up and be sent to the functional units within a single clock cycle in order to maintain high processor throughput. Conventional issue queues have a CAM/RAM structure where the CAM holds the source operand tags and the RAM holds the remaining information. Each entry has a valid-bit to indicate its status. The valid-bit is set when the entry is allocated for a dispatched instruction and is reset when the instruction is issued and leaves the issue queue. We put invalid entries into the recovery boost mode. The CAM performs tag-matching operations against all the broadcasted tags each clock cycle. In order to do this, each CAM entry has a set of comparators and matchlines and the number of these required depends on the issue width of the processor. In each cycle, each matchline is precharged. If there is a mismatch between the tag data in the memory cell and the broadcasted result tag in any of the CAM cells in the issue queue entry, then the corresponding matchline is discharged; otherwise the matchline stays high. If any of the matchlines for a given operand tag entry stays high, its corresponding ready signal is asserted high via a OR-block.

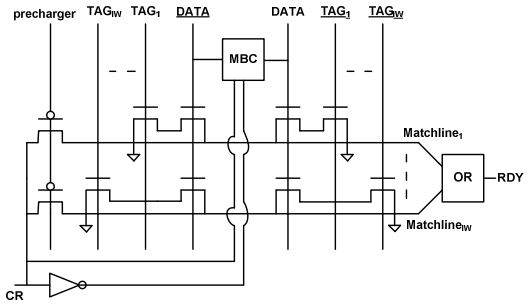


Fig. 4. Modified CAM Structure (IW = issue width). MBC is the Modified Bit-Cell for recovery boosting.

To provide recovery boosting, the memory cells of the RAM and CAM structures are composed of the modified SRAM cells shown in Figure 2(a). The modified issue queue entry is shown in Figure 4. The valid-bit works as the control signal (CR) for the entry. When the memory cells in the entry transition to the valid state, the CR signal becomes high which pulls the ground down to low and the entry works in the normal operating mode. When the entry transitions to the invalid state, the CR signal becomes low and puts the memory cell into the recovery boost mode. When in the recovery boost mode, both nodes of the memory cells in both the RAM and the CAM parts are raised to V_{dd}. Due to the high node voltages, the comparators in the CAM will be triggered leading to a discharge of the matchline. To avoid this unnecessary precharging and discharging of the matchline (which wastes power), we further modify the issue queue entry so that the prechargers of the matchlines are connected to the CR signal. When CR is high in the normal operating mode, the matchlines will be precharged to V_{dd} and the tag-matching process will continue each cycle. When CR is low during the recovery boost mode, the matchlines stay low and therefore do not discharge.

IV. Circuit-Level Simulation Results

We perform SPICE-level simulation using the Cadence Spectre circuit simulator to verify the functionality of our designs and determine their area and power consumption. Our experiments are carried out for 32nm process using the Predictive Technology Model. Our bitcell device sizes are: PMOS = 58nm $\times$ 33nm, NMOS = 87nm $\times$ 33nm, Access Transistor = 58nm $\times$ 41nm. We simulate two designs of the issue queue: the baseline design that uses conventional 6T SRAM cells (which do not provide recovery boosting) and the design that uses the modified SRAM cells discussed in Section II to provide fine-grained recovery boosting.

• ***Functionality:*** We verified functionality through transistor-level simulation of the issue queue. We evaluated whether we can perform read, write, and hold operations on the modified SRAM cells and whether the modified cells correctly switch between the normal operating modes and the recovery boost mode. We took into account the extra inverter delay required for changing the ground voltage of the cell during these transitions. We verified that these memory structures operate correctly in the normal operating modes and also that writes are performed correctly to the cells when transitioning out of the recovery boost mode.

Clock Frequency Setting: In our simulations, we found the smallest possible cycle-time for the modified SRAM cell to be 220ps (a clock frequency of 4.5 GHz). We choose a more conservative cycle-time of 333ps, which corresponds to a clock frequency of 3 GHz. We found the delay of the high-V_t access transistors that connect to the V_{dd} rail to be small enough to transition the cell into the recovery boost mode within a single cycle for the 3 GHz clock frequency.

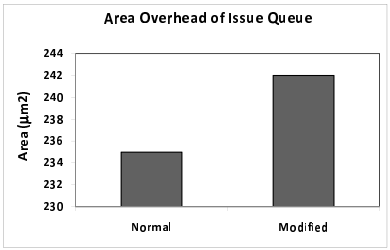


Fig. 5. Area of the the issue queue for designs that use conventional 6T cells and modified cells for recovery boosting.

• ***Area:*** We designed the issue queue for both the baseline and recovery boosting cases to occupy the minimum area required to provide correct functionality. Care was taken to size the devices so that they are of minimal size while meeting the 3 GHz clock frequency requirement. The overheads for the multi-ported issue queue is given in Figure 5.

Since the issue queue has 4 read and write-ports respectively, each bitcell has 16 transistors: 4 transistors for the inverter-pair, 8 transistors for the write-ports, and 4 transistors for the read-ports (for supporting single-ended reads). To support recovery boosting, we add 2 extra transistors of minimal size to each cell and one extra inverter for an entire row of 65 bitcells. Therefore, adding the extra transistors for recovery boosting to this heavily multi-ported structure is expected to add only a small amount of area. Indeed, we can see that the area of issue queue that uses the modified

cells is only 3% more than the baseline design. This overhead is roughly equivalent to the area occupied by two entries in the modified issue queue. We can therefore design the issue queue to be area-neutral with respect to the baseline (i.e., occupy the same area as the baseline design) by having its capacity reduced by two entries. The performance impact of this area-neutral design is evaluated in Section VI.

• ***Dynamic and Leakage Power Consumption:*** Figure 6 gives the power consumption of a single issue queue entry for both the baseline design and the one that uses the modified SRAM cell. For the issue queue entry, in addition to the power consumed in the recovery boost mode, we quantify the power consumed in each of the three normal operating modes (read, write, and hold). For each of these modes, we present the power consumption for two scenarios: (i) when both source tags of an entry mismatch with the ones broadcast down the issue queue in the same cycle, which is the highest power consumption scenario since all the matchlines discharge, and (ii) when both source tags match in the same cycle, which consumes the least amount of power.

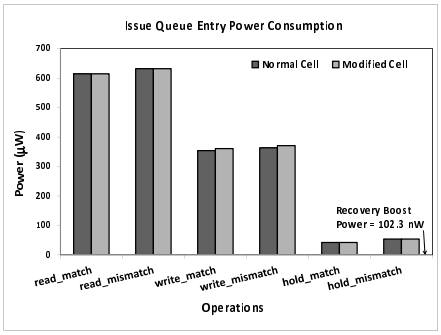


Fig. 6. Power consumption of a single issue queue entry.

We can see that the power consumed by the designs that use the modified SRAM cells for the read, write, and hold operations are nearly equal to those of the baseline designs. The maximum increase in power is less than 2% for the issue queue entry. The slight increase in power for the recovery boost designs is due to leakage in one of the PMOS access transistors that connect to the V_{dd} rail. The sources of the PMOS access transistors are connected to V_{dd} and the drains are connected to the nodes. Therefore, based on whether a cell holds a '0' or a '1', one of the two PMOS devices will leak. Since we use high-V_t PMOS devices as the access transistors for the cells (to reduce the impact of NBTI), the leakage power of these transistors is also reduced.

In memory arrays that use conventional SRAM cells, the cells will normally be operating in the hold mode when they house invalid data. However, when the modified cells are used, cells that hold invalid data can operate in the recovery boost mode. We can see that the power consumed in the recovery boost mode is orders of magnitude less than in the hold mode. This is because the recovery boost operation raises Node0 and Node1 (shown in Figure 2(a)) and the ground to V_{dd}, which cuts off the path from V_{dd} to ground and significantly reduces the leakage currents. Finally, there is a small power benefit at the structure level since we use an area-neutral design for the issue queue which is slightly smaller than the baseline design.

V. Architectural Analysis Methodology

We carry out our architecture-level evaluations via execution-driven simulation using the M5 simulator [2]. Our workloads consist of benchmarks from the SPEC CPU2000 benchmark suite [14]. We perform detailed simulation of the first 100-million instruction SimPoint for each benchmark [11]. We present simulation results for 16 representative benchmarks - 8 integer and 8 floating-point. We model a 4-wide issue core, which is similar to those in multicore 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 7 years based on the work by Tiwari and Torrellas [15].

Static Noise Margin(SNM): NBTI can affect read and write times of the SRAM cells as well as their read SNM (SNM). Previous work [6] 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 workloads, the bitcells in 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 the PMOS devices in the issue queue (based on our circuit-level designs) over the course of an architecture simulation and extrapolate the statistics to calculate the degradation in V_t after the 7-year service life. We then feed the V_t values of these PMOS devices into the Spectre circuit simulator to calculate the SNM of all the cells in a structure at the end of the 7-year period and use the smallest value to denote the SNM for that structure.

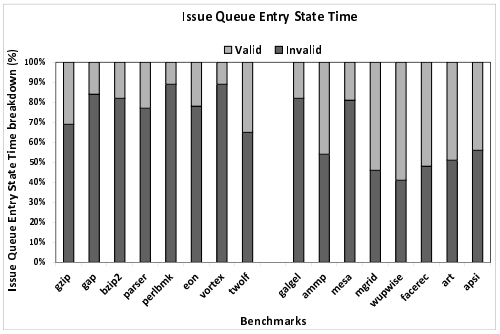
VI. Architecture-Level Simulation Results

We now study the impact of putting memory cells of the issue queue entries into the recovery boost mode when they hold invalid data. We evaluate three different processor configurations, which we denote as: *Baseline*, *Recovery Boosting*, and *Balancing*. *Baseline* models 4-wide issue core that does not use any NBTI mitigation technique. *Recovery Boosting* replaces the issue queue of the baseline configuration with its counterpart that supports recovery boosting. We assume that the modified issue queue is designed to be area-neutral with respect to *Baseline* by trading off a small amount of storage capacity to accommodate the extra area required to implement recovery boosting. Based on our area evaluations in Section III, we assume that the modified issue queue has 62 entries. In all our simulations, we find that this reduction in capacity has a negligible impact on performance and therefore we do not present detailed performance results. *Balancing* denotes a recovery enhancement scheme similar to the one proposed in [1] that uses the same time intervals that *Recovery Boosting* exploits to balance the degradation of the two PMOS devices in the memory cell. As pointed out by Abella et al. [1], flipping the contents of the memory cells only when they hold invalid data instead of when they hold both valid and invalid data, for which additional circuitry is required [6], is the preferable approach for high-speed SRAM structures in order to not increase their delay significantly. Since the access time of the issue queue has a strong impact on processor performance, the *Balancing* technique is applied only when the memory cells hold invalid data. We optimistically assume that *Balancing* does not impose any additional area overheads over the baseline design and that it can keep the inputs to each PMOS device at a logic '0' exactly 50% of the time whenever the cells are in this mode.

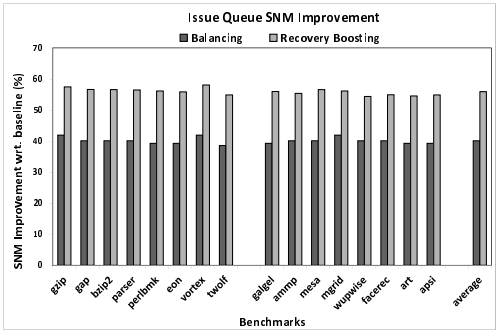
The breakdown of the time spent by the issue queue entries in the *Valid* and *Invalid* states for each benchmark is shown in Figure 7(a) and the corresponding reliability results are

given in Figure 7(b). The time-breakdown in Figure 7(a) is an average over all the entries in the issue queue and over the entire SimPoint of each benchmark.

As the Figure shows, while *Balancing* provides a good improvement in the SNM, *Recovery Boosting* provides significantly higher reliability benefits by virtue of its ability to put both PMOS devices into the recovery mode. Across all the benchmarks, *Balancing* provides a 40% improvement in the SNM while *Recovery Boosting* provides a 56% improvement. These results clearly highlight the benefits of recovery boosting as a technique to mitigate NBTI in the issue queue.



(a) Breakdown of time spent by the issue queue entries in the *Valid* and *Invalid* states.



(b) Improvement in the SNM over the baseline processor configuration.

Fig. 7. Reliability Behavior of the Issue Queue.

VII. Related Work

There are two basic approaches to mitigating NBTI: (i) reduce the stress on the PMOS transistors; (ii) 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. Recovery boosting is a recovery enhancement technique for SRAM structures.

Stress reduction techniques: Tiwari and Torrellas propose a technique called "Facelift" [15] to hide the effects of aging through temperature-based job-scheduling to individual cores of a multicore processor, in conjunction with V_{dd} and V_t control. The use of stress reduction techniques is orthogonal to the use of recovery enhancement.

Recovery enhancement techniques: Abella et al. [1], Kumar et al. [6], and Shin et al. [12] propose techniques to balance the degradation of the two PMOS devices in the memory cells 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 US Patent describes an idea similar to coarse-grained recovery boosting using a cell design similar to the *Alt-Cell-1* alternative cell discussed in Section II [3]. However, the idea in the patent has the same drawback as *Alt-Cell-1*: it takes multiple processor clock cycles to put both PMOSes into recovery and therefore cannot be used for high-speed SRAM arrays. Recovery boosting, on the other hand, can be used for high-speed arrays and can be used with both fine- and coarse-grained control. Moreover, the patent does not provide any analysis of the impact (qualitatively nor quantitatively) of using their technique to design microarchitectural structures.

VIII. Conclusion

NBTI is an important silicon reliability problem. SRAM memory cells are especially vulnerable to NBTI. We propose recovery 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 design and evaluate an issue queue, at both the circuit and architecture levels, that uses recovery boosting and show that this technique is effective in providing NBTI recovery and is efficient in terms of area, power, and performance.

Acknowledgment

We thank Mircea Stan and Adam Cabe for their valuable inputs. This research has been supported in part by NSF grant 0627527 and gifts from Intel.

References

[1] J. Abella, X. Vera, and A. Gonzalez. Penelope: The NBTI-Aware Processor. In *Proceedings of the 40th IEEE/ACM International Symposium on Microarchitecture*, 2007.

[2] N.L. Binkert and et al. The M5 Simulator: Modeling Networked Systems. *IEEE Micro*, 26(4), July 2006.

[3] P. Bose, J. Shin, and V. Zyuban. Method for Extending Lifetime Reliability of Digital Logic Devices Through Removal of Aging Mechanisms, February 2009. US Patent 7,489,161.

[4] Cadence Virtuoso Spectre Circuit Simulator. http://www.cadence.com/products/cic/spectre_circuit/.

[5] D. Folegnani and A. Gonzalez. Energy-Effective Issue Logic. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 2001.

[6] S.V. Kumar, C.H. Kim, and S.S. Sapatnekar. Impact of NBTI on SRAM Read Stability and Design for Reliability. In *Proceedings of the International Symposium on Quality Electronic Design*, 2006.

[7] S. Palacharla. *Complexity-Effective Superscalar Processors*. PhD thesis, University of Wisconsin - Madison, 1998.

[8] Predictive Technology Model. http://www.eas.asu.edu/~ptm/.

[9] G. Reimbold and et al. Initial and PBTI-induced traps and charges in Hf-based oxides/TiN stacks. *Microelectronics Reliability*, 47(4-5), April 2007.

[10] D.K. Schroder and J.A. Babcock. Negative Bias Temperature Instability: Road to Cross in Deep Submicron Silicon Semiconductor Manufacturing. *Journal of Applied Physics*, 94(1), July 2003.

[11] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, October 2002.

[12] 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 *Proceedings of the International Symposium on Computer Architecture*, 2008.

[13] A. Sil, S. Ghosh, N. Gogineni, and M. Bayoumi. A Novel High Write Speed, Low Power, Read-SNM-Free 6T SRAM Cell. In *Proceedings of the Midwest Symposium on Circuits and Systems (MWSCAS)*, August 2008.

[14] SPEC CPU2000. http://www.spec.org/cpu2000/.

[15] A. Tiwari and J. Torrellas. Facelift: Hiding and Slowing Down Aging in Multicores. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*, November 2008.

[16] X. Yang, E. Weglarz, and K. Saluja. On NBTI Degradation Process in Digital Logic Circuits. In *Proceedings of the International Conference on VLSI Design*, January 2007.