Improving The Efficiency Of SRAM Array Using Recovery Boosting Technique To Reduce The Effect Of NBTI

Kumar Neeraj[1], Hari Shanker Srivastava[2]
Anurag group of institutions, Hyderabad

Abstract: - Negative bias temperature instability (NBTI) is an important lifetime reliability problem in microprocessors. The degradation of PMOS devices due to NBTI leads to reduced temporal performance in digital circuits. We have analysed the impact of NBTI on SRAM cells. The SRAM exhibit a major problem called negative bias temperature instability while storing the data ‘0’ so large amount of energy is wasted here. However, one of the devices is always in the negative bias condition at any given time. In existing system recovery techniques for SRAM cells aim to balance the degradation of the two PMOS devices by attempting to keep their inputs at logic “0” exactly 50% of the time. However, one of the devices is always in the negative bias condition at any given time. The negative bias can lead to the generation of interface traps at the Si/SiO2 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. In this paper, we propose a simple solution to recover the SRAM cell using a recovery boosting technique and present the results simulated on DSCH & MICRO WIND. We also compare and evaluate different implementation methodologies for the proposed technique. 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 show that Recovery Boosting provides significant improvement in power consumption and performance. In our paper we are going to remove this hazard & we are going to modify the SRAM circuit using Recovery Boosting. As well as we are also going to design SRAM array for improving efficiency as well as read and write stability of the circuit.

Keywords: Nbti, Recovery Boosting, Sram, Microwind

1. 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[1-3]. 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/SiO2 interface, which cause an increase in the threshold voltage of the device.[4-6]. 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. 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.[7]. 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. Before the end of 2003, microprocessors were able to produce 90 nm processes, and now production on 45 nm and below is coming out.[8-9]. However, the scaling of CMOS technology into deep submicron regimes has brought about new reliability challenges in MOSFET device such as hot carrier Injection (HCI), Negative Bias Temperature Instability (NBTI),Time Dependent Dielectric Breakdown (TDDB), radiation induced damage, etc., which can pose a limit to the device scaling, and cause circuit performance degradation.[10].

When integrate circuits work for long time, especially for today’s CMOS technology, heat of chips will increase significantly. For this reason, one of the dominant reliability issues - Negative Bias Temperature Instability (NBTI) in PMOS transistors will be studied in this work. NBTI impact gets even worse in scaled technology due to higher operation temperature and the usage of ultra thin oxide (i.e., higher oxide field).[11].

2. RELIABILITY ANALYSIS

One of the major temperature-induced reliability issues for p-channel metal-oxide semiconductor field effect transistors (PMOSFETs) is the negative bias temperature instability (NBTI), which is caused by stressing negative gate bias at elevated temperatures. A positive charge builds up at the channel interface of PMOSFETs under negative bias and high temperature conditions. The typical temperature conditions for NBTI effects are in the range of 100 ~ 250 °C , and the oxide electric fields should be below 6MV/cm. During NBTI–induced device degradation, the holes in the inversion layer react with the hydrogen-silicon bonds (Si-H) at the Si/SiO2 interface and then release the hydrogen species(atom, molecule or ion) by breaking the Si–H bonds, because the two-electron Si-H covalent bond is weakened. Once a hole is captured and this weakened bond is easily broken at relatively moderate temperature. The electrochemical reactions at the SiO2/Si interface are given as follows

NBTI is a key reliability issue in MOSFETs. It is of immediate concern in p-channel MOS devices, since they almost always operate with negative gate-to-source voltage; however, the very same mechanism affects also NMOS transistors when biased in the accumulation regime, i.e. with a negative bias applied to the gate too. NBTI manifests as an increase in the threshold voltage and consequent decrease in drain current and trans conductance. The degradation exhibits logarithmic dependence on time.

In the sub-micrometer devices nitrogen is incorporated into the silicon gate oxide to reduce the gate leakage current density and prevent the boron penetration. However, incorporating nitrogen enhances NBTI. For new technologies (32 nm and shorter nominal channel lengths), high-K metal gate stacks are used as an alternative to improve the gate current density for a given equivalent oxide thickness (EOT). Even with the introduction of new materials like hafnium oxides, NBTI remains.

3. WHAT IS THE NBTI EFFECT?

The semiconductor process evolution that produces small transistors increases the potential for interface traps in PMOS transistors during prolonged times of negative bias stress. An interface trap is created when a negative voltage is applied to the gate of a PMOS device for a prolonged time (see Figure 1). An interface trap is located near the Si-oxide/Si-crystal lattice boundary where holes (positive charge) can get stuck, and in doing so, they shift the threshold voltage. This hole trapping creates interface states as well as fixed charges. Both are positive charges and result in a negative shift of threshold voltage. This phenomenon is called PMOS Negative Bias Temperature Instability (NBTI). NMOS transistors are far less affected because interface states and fixed charges are of opposite polarity and eventually cancel each other.

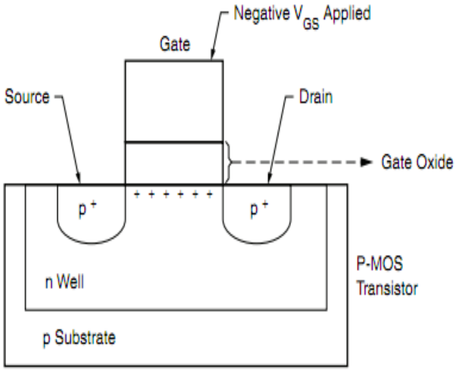


Fig 1: PMOS Transistor

NBTI can determine the useful lifetime of complementary metal-oxide semiconductor (CMOS) devices. The designer must consider the bias conditions of each PMOS transistor, not only at the beginning of life but throughout the expected lifetime of the product. Typically, these conditions must allow for at least ten years of operation at the highest voltage and the highest temperature. Removal of power allows for some release (annealing) of the trapped charges. Static stress shifts the voltage threshold roughly ten times more than does dynamic stress. For example, a shift of 10 mV can occur in a dynamic (switching) situation, and a shift of 100 mV can occur in the static case. When silicon is oxidized, most of the Si atoms at the surface of the wafer bond with oxygen while a few atoms bond with hydrogen. When a negative bias (i.e., a logic input of “0”) is applied at the gate of a pMOS transistor, the relatively weak Si-H bonds get disassociated, leading to the generation of interface traps at the Si/SiO2 interface. These interface traps cause the threshold voltage of the pMOS transistor to increase, which in turn degrades the speed of the device and the noise margin of the circuit, eventually causing the circuit to fail [11]. A detailed discussion on the physics of interface trap generation due to NBTI is given in. The period of time when the pMOS transistor is negatively biased is known as the stress phase or stress mode. The increase in due to stress is given by the equation

$$\Delta V_{\mathrm{ts}} = \left(\frac{q t_{\mathrm{ox}}}{e_{\mathrm{ox}}}\right)^{3/2} \cdot K_1 \cdot \sqrt{C_{\mathrm{ox}}(V_{\mathrm{gs}} - V_t)} \cdot e^{-E_a/4kT + 2(V_{\mathrm{gs}} - V_t)/t_{\mathrm{ox}} E_{01}} \cdot T_0^{-0.25} \cdot t_{\mathrm{stress}}^{0.25}$$

where is the time under stress, is the oxide thickness, and is the gate capacitance per unit area.and are constants equal to C nm , 0.49 eV, s/nm , 0.08 V/nm, and eV/K, respectively. When a logic input of “1” is applied at the gate, the transistor turns off eliminating some of the interface traps. This is known as the recovery phase or recovery mode. The final increase in after considering both the stress and recovery phases is given by

$$\Delta V_t = \Delta V_{\mathrm{ts}} \cdot \left(1 - \frac{2\xi_1 t_{\mathrm{ox}} + \sqrt{\xi_2 e^{-E_a/kT} T_0 t_{\mathrm{rec}}}}{(1+\delta) t_{\mathrm{ox}} + \sqrt{e^{-E_a/kT}(t_{\mathrm{stress}} + t_{\mathrm{rec}})}}\right)$$

From the equations, one can observe several options for reducing the impact of NBTI. We can see that NBTI is affected by temperature, the stress and recovery times and the difference between and Reducing, temperature, the stress time and increasing can reduce the stress on the transistors whereas longer recovery times enhance the recovery process. However, using a lower (which also provides a reduction in temperature) can reduce performance. Similarly, although a higher device is likely to be more resilient against NBTI, such devices are slower than their lower counterparts. In this paper, we tackle the NBTI problem by increasing the recovery time..

4. RECOVERY-ENHANCEMENT TECHNIQUES:

Abella et al. propose 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. propose a similar technique to periodically flip the contents of SRAM cells to balance the wear on the pMOS transistors. Shin et al. propose a recovery 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 the pMOS devices to put them into a deep recovery state. When an array is put into the recovery mode, the pMOS devices in one of the inverters in all of the cells are put into the recovery mode followed by those in the other inverter and this recurring pattern is continued throughout the recovery period for the array. All of these recovery enhancement techniques aim to balance the degradation 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. 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, and two access transistors. 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 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 for read or write, it is expected to hold the data stored in it and is said to operate in the hold mode.

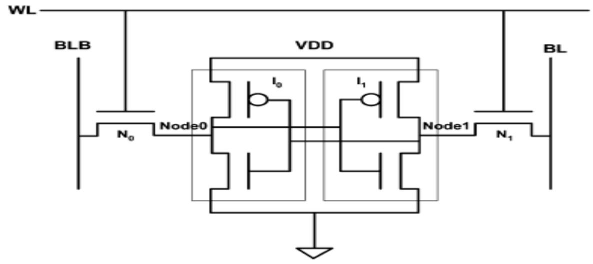


Fig 2: Basic SRAM cell

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.

We propose a 6T SRAM cell design shown in Fig. 2 which is capable of normal operations (read, write, and hold) as well as providing an NBTI recovery mode (when the cell does not contain valid data) that we call the recovery boost mode where both pMOS devices within the cell undergo recovery at the same time. We refer to the period when the cell does not contain valid data that is never used by any other micro architectural structure in the processor as “invalid period.” The basic idea behind recovery boosting is to raise the node voltages (Node0 and Node1 in Fig. 2) of a memory cell in order to put both pMOS devices into the recovery mode.

This can be achieved by raising the ground voltage to the nominal voltage through an external control signal. The modified SRAM cell has the ground connected to the output of an inverter, as shown in Fig.3. 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”, 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. 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 word line.

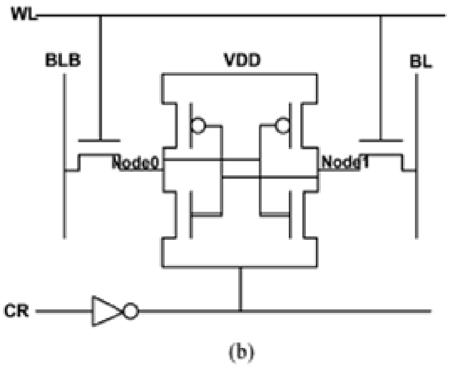


Fig 3: Modified SRAM cell

However, the drawback of this approach is that it can take a long time to raise both the node voltages to in a high-performance processor that operates at a high clock frequency. This is illustrated in Fig.4 , which presents the achieved pMOS gate voltages of a bit cell over time due to recovery boosting. Model.3 The operating temperature is 90 C, which is the average temperature in which the high-performance processors operate. We use this temperature value throughout the paper for all the experiments. We can observe that this approach achieves the desired gate voltage within 3.33 ns. For a processor which operates at 3-GHz frequency, it will take ten cycles to switch to the recovery boost mode.

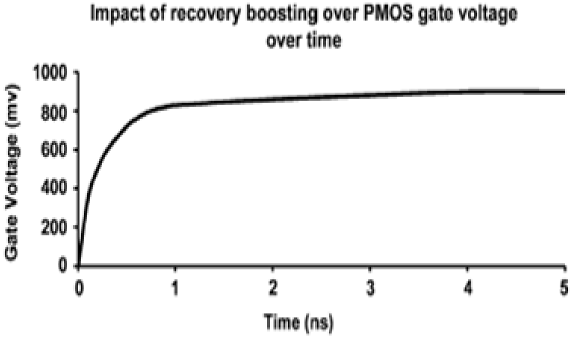


Fig 4: PMOS gate voltages of SRAM cell

Similarly, it takes around ten cycles to go back to the normal operating mode from the recovery boost mode. However, our goal is to be able to switch between 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 mentioned before, recovery boost mode is applied when an entry of the structure holds data that is considered “invalid” at the architecture- 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.4.2 will take 20 cycles of the 50 cycles (40% of the invalid period) to shift between 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 negative 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. There are various ways of incorporating such cells into SRAM arrays, which we will discuss shortly. 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. We now discuss how the modified high-speed recovery boosting SRAM cells can be used in each of these scenarios and then discuss additional microarchitectural issues related to implementing recovery boosting.

5. DESIGNING MICRO ARCHITECTURAL STRUCTURES THAT SUPPORT RECOVERY BOOSTING

Having discussed the basics of recovery boosting, we now turn our attention to designing SRAM-based micro architectural structures that use this technique to provide resilience against NBTI. Here, we present and evaluate the circuit-level design of two large SRAM-based structures within a four-wide issue processor core, namely, the physical register file and the issue queue, which we modify to support recovery boosting. We study the design of a 128-entry multi ported physical register file with eight read-ports, four write-ports, and 64-b entries. The issue queue uses a non-data-captured design and consists of 64 entries with four read-ports, four write-ports and 65 b per entry. The choice for the entry size is based on the issue queue descriptions. The mappings from architected registers to physical registers are maintained in a register alias table (RAT). The physical register is returned to the free list of registers when the next instruction that writes to the same architected register commits. A physical register goes through a sequence of four states, shown in Fig. 5: 1) it is not mapped to any producer instruction and is free (Unmapped); 2) it is mapped to an instruction but it has not yet been written into by that instruction (Mapped- Invalid); 3) it holds a valid value that has been written to it (Mapped-Valid); and 4) it holds a valid value but the value is not read by any instruction before it is released to the pool of free registers (Post Last-Read). Once the register completes thePost Last-Read state, the register returns to the Unmapped state and remains in that state till the register-renaming logic chooses it again for a mapping.

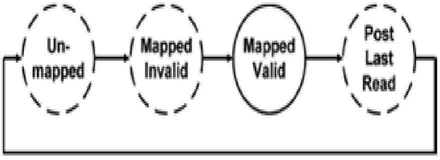


Fig 5: Register states

There are three candidate states that one could use for recovery boosting: Unmapped, Mapped-Invalid and Post Last-Read. When a physical register is in the Unmapped and Mapped-Invalid states, it does not hold valid data and therefore we can put its cells into the recovery boost mode without affecting architecture l correctness. The cells will need to be transitioned into the normal operating mode when moving from the Mapped-Invalid to the Mapped-Valid state, which occurs then the producer instruction has completed its execution and forwards the value to the register file. Post Last-Read is a more complex situation. Several cycles may elapse between the last time that the physical register is read and when the register is released. This time period could potentially be exploited for recovery boosting but it is challenging to precisely determine when the last-use of a register is complete. Although there have been proposals to exploit this time window to speculatively release registers early for the sake of performance and power queue uses a non-data-captured design and consists of 64 entries with four read-ports, four write-ports and 65 b per entry. The choice for the entry size is based on the issue queue descriptions given in. Post Last-Read is a more complex situation. Several cycles may elapse between the last time that the physical register is read and when the register is released. This time period could potentially be exploited for recovery boosting but it is challenging to precisely determine when the last-use of a register is complete. Although there have been proposals to exploit this time window to speculatively release registers early for the sake of performance and power queue entry to be similar to the one described by Palacharla et al. 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 valid-bit itself is not put into the recovery boost mode to ensure correct operation of the instruction scheduler). 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 the number of comparators required depends on the issue width of the processor. The design of the CAM part of the issue queue for a single bit is shown in Fig. 6. In each cycle, each match line is pre-charged. 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 match line is discharged; otherwise the match line stays high. If any of the match lines for a given operand tag entry stays high, its corresponding ready signal (RDY) is asserted high via the OR-block shown in the figure.

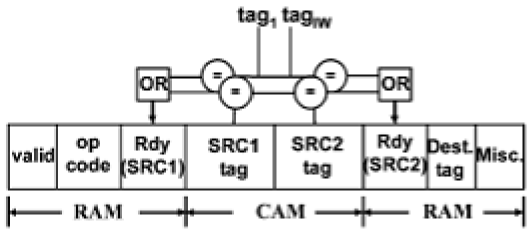


Fig 6: Register Queue entry

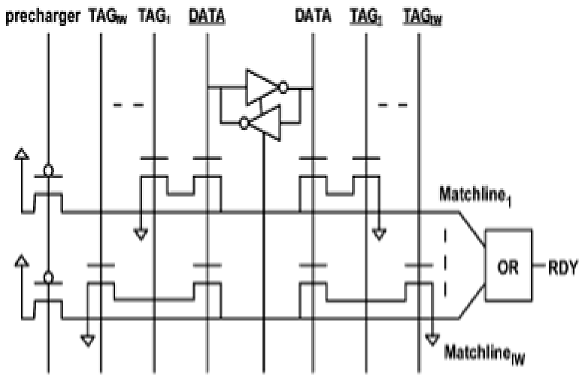


Fig 4.9: CAM structure if issue queue entry

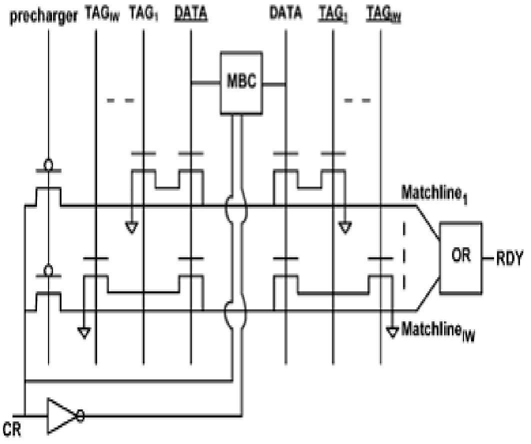


Fig 7: Modified CAM structure

The modified issue queue entry is shown in Fig.7. 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. Due to the high node voltages, the comparators in the CAM will be triggered leading to a discharge of the match line. To avoid this unnecessary pre charging and discharging of the match line(which wastes power), we further modify the issue queue entry so that the pre chargers of the match lines are connected to the CR signal. .

In Fig. 8, we show that the node voltages (Node0 and Node1) of the modified cell for recovery boosting takes 160 ps to reach desired voltage values whereas, the conventional 6T cell takes 140 ps. Even though the write delay is increased by 20 ps because of the increased capacitance in the modified cell, this operation can be done within a single cycle of a high performance processor. Since the read and hold operations behave in a similar way for both modified and conventional 6T cells, we do not present the waveforms for these operations.

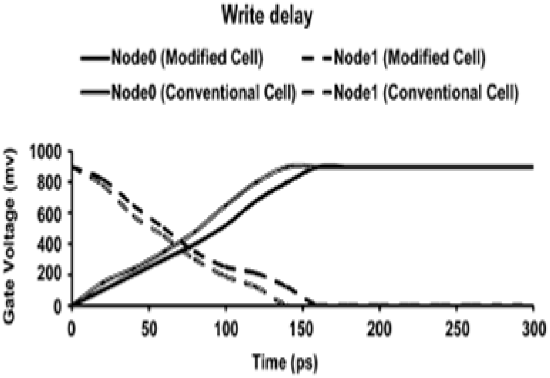


Fig 8:write delay of Modified SRAM cell

Fig shows that the required time to switch to recovery boost mode is 190 ps. To switch to the recovery boost mode, both node voltages have to rise to. Node0 stays in and Node1 takes 190 ps to rise to. Since a bitcell transitions from recovery boost mode to normal operating mode on a write, we ran simulation to confirm that we can successfully write to the cell after the transition within a cycle. Fig.9 shows the required time to switch to normal operating mode from recovery boost mode with a write operation. In recovery boost mode, both node voltages (Node0 and Node1) stays in . Therefore, with a write operation, Node0 has to stay in and Node1 has to be pulled down to GND. As we can see from the figure, Node1 reaches the desired value (GND) within 140 ps. therefore, shifting to the recovery boost mode and to come back to the normal operating mode from it take 190 and 140 ps, respectively. Fig

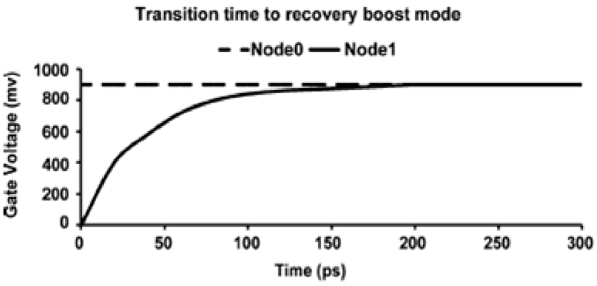


Fig 9: Transition to recovery boost mode

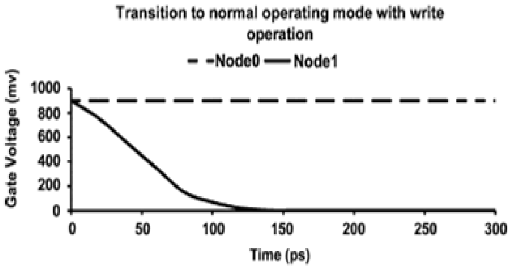


Fig 10: Transition to normal operating mode

.

VII.IMPLEMENTATION AND RESULTS

Basic SRAM cell:

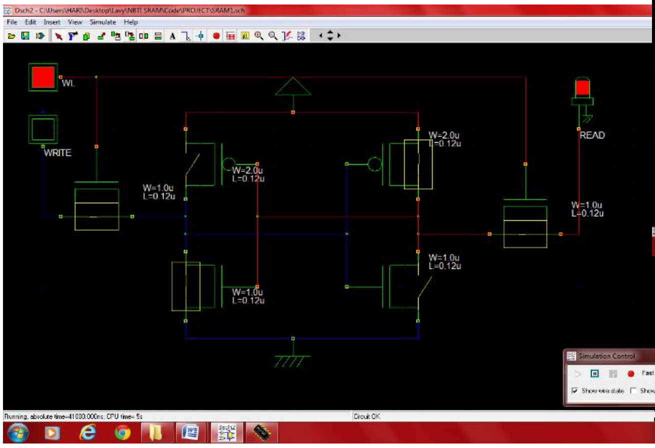


Fig 11: basic SRAM cell working

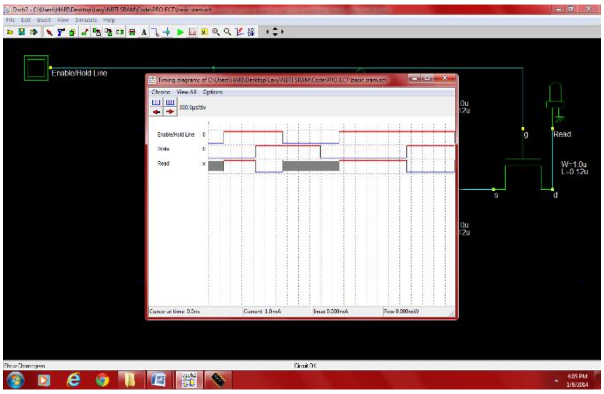


Fig 12: simulations result of SRAM Cell

Layout:

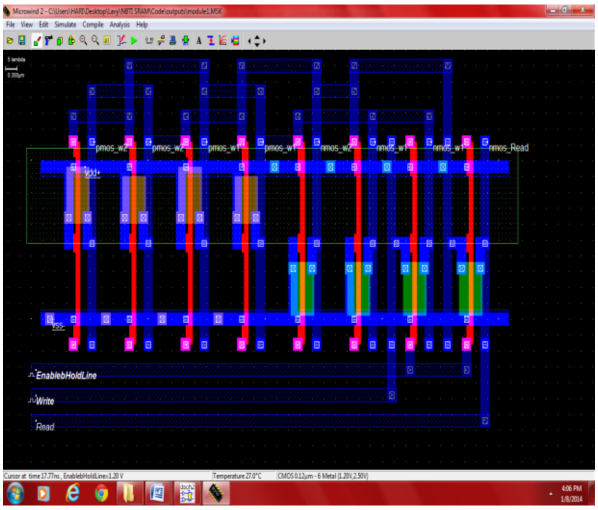
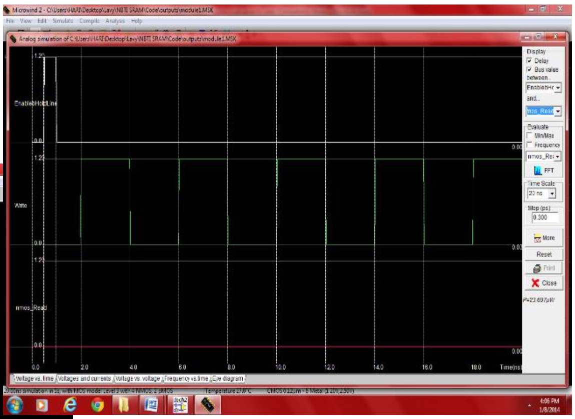


Fig 13: LAYOUT for basic SRAM cell



ig 14: layout results for basic SRAM cell

SRAM with CR:

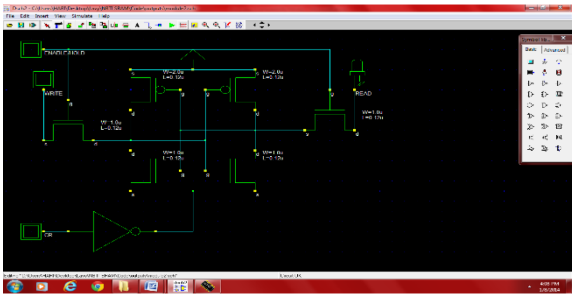


Fig 15: SRAM cell with CR signal

When CR is high:

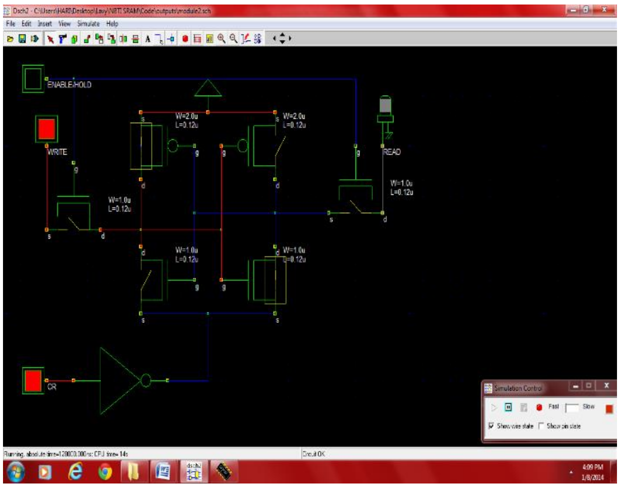


Fig 16: working of SRAM cell with CR

Output:

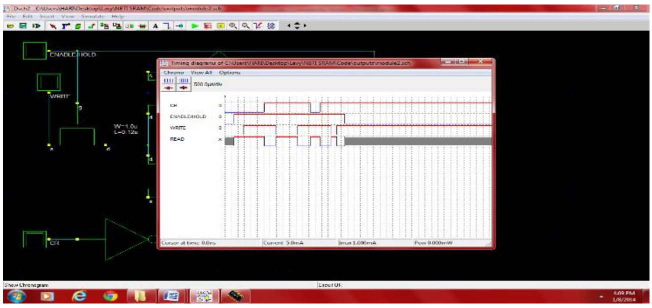


Fig 17: TIMING ANALYSIS of sram cell with CR

Layout for SRAM with CR:

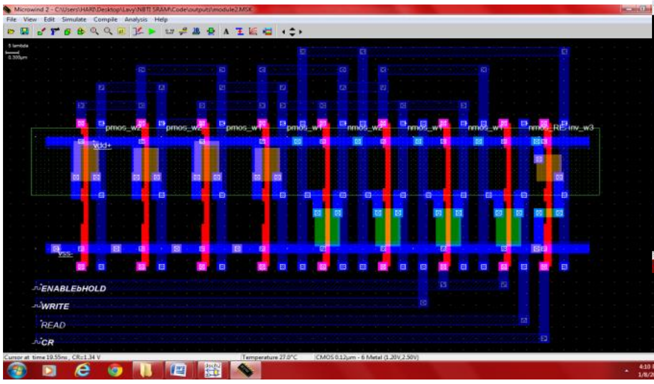


Fig 18: Layout of SRAM cell with CR

Recovery Boosting circuit:

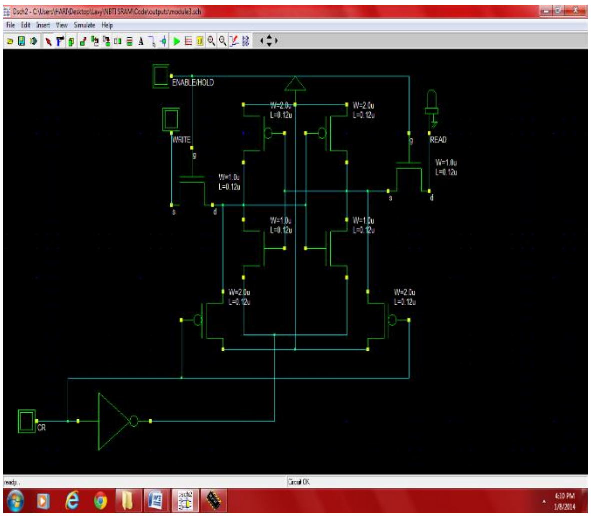


Fig 19: Design of recovery boosting circuit

Working of Recovery Boosting circuit:

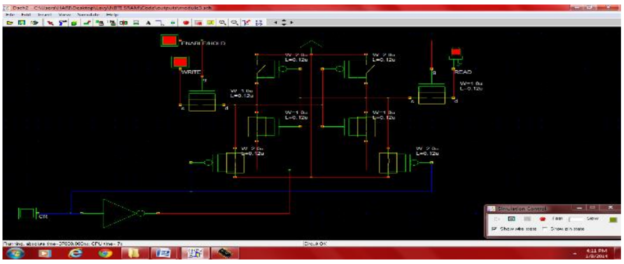


Fig 20: Working of recovery boosting circuit

Simulation Results:

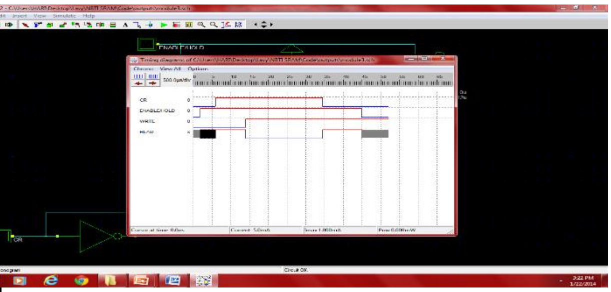


Fig 21: Timing analysis of recovery circuit

Layout:

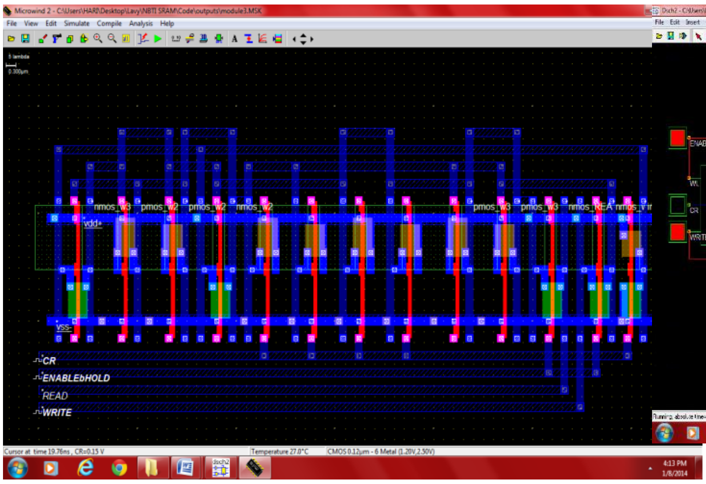

Microwind 2 - C:\Users\HARI\Desktop\Lavy\NBTI SRAM\Code\outputs\module1.MSK
File View Edit Simulate Compile Analysis Help
nmos_w3
pmos_w2
pmos_w2
nmos_w2
pmos_w3
pmos_w3
nmos_REA
nmos_Vr
vdd+
vss-
CR
ENABLEbHOLD
READ
WRITE
Cursor at time 19.78ns , CR=0.15 V
Temperature 27.0°C
CMOS 0.12µm - 6 Metal (1.20V,2.50V)
4:13 PM
1/8/2014

Fig 22: Layout for Recovery boosting circuit

Layout results:

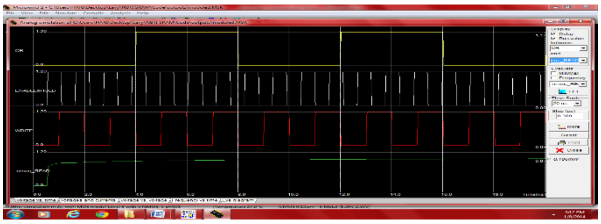


Fig 23: Layout results of recovery boosting circuit

4-BIT SRAM:

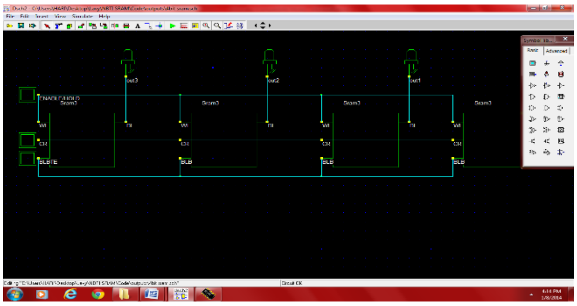


Fig 24: Schematic design of 4-bit SRAM array

How 4-BIT SRAM works:

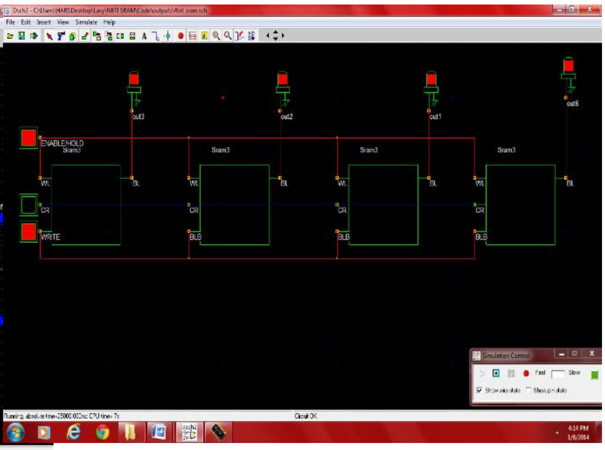


Fig 25: Working of 4-bit SRAM ARRAY

SIMULATION RESULT:

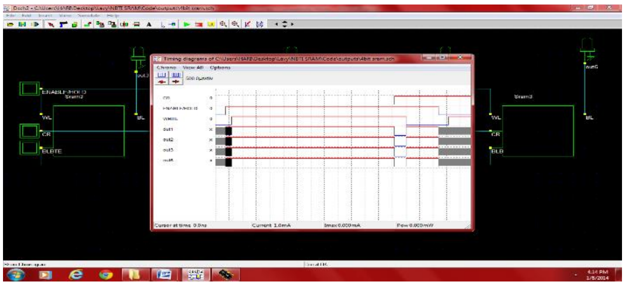

Timing diagrams of C:\Users\HARI\Desktop\Lavy\NBTI SRAM\Code\outputs\4-bit sram.sch
Chrono
View All
Options
CR
ENABLE/HOLD
out1
out2
out3
Sram3
WL
BL
CR
BLB

Fig 26: Timing analysis of 4-bit SRAM

4-BIT SRAM Layout:

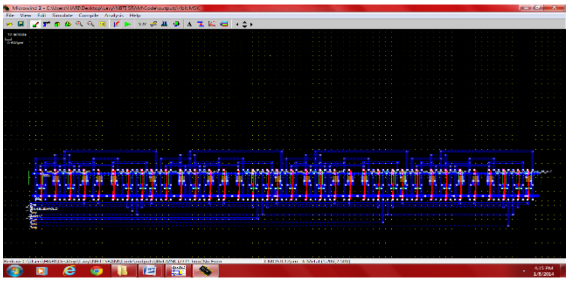

Microwind 2 - C:\Users\HARI\Desktop\Lavy\NBTI SRAM\Code\outputs\4-bit.MSK

Fig 27: Layout design of 4-bit SRAM Array

Post layout simulation result:

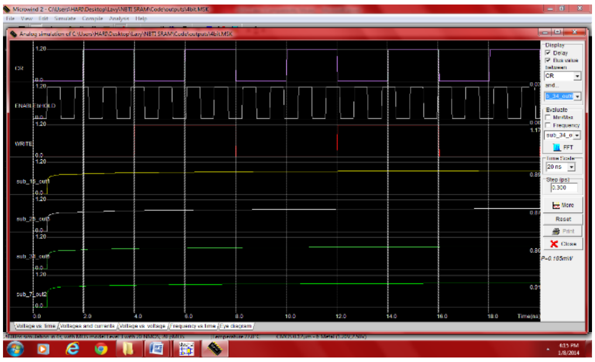
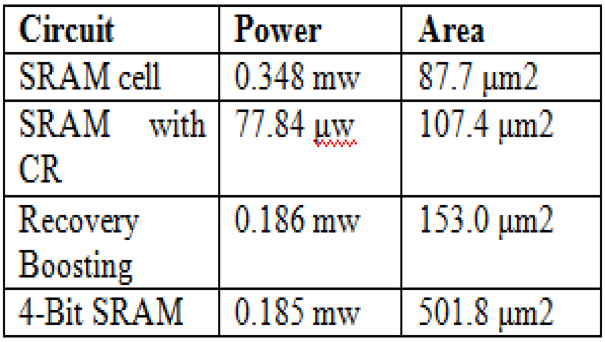


Fig 28: Layout results of 4-bit SRAM array

COMPARISION TABLE:



Circuit	Power	Area
SRAM cell	0.348 mw	87.7 µm2
SRAM with CR	77.84 µw	107.4 µm2
Recovery Boosting	0.186 mw	153.0 µm2
4-Bit SRAM	0.185 mw	501.8 µm2

Table 1: comparison between SRAM CELL and modified SRAM cells

6. CONCLUSION AND FUTURE ENHANCEMENT

NBTI is one of the most important silicon reliability problems facing rocessor designers. SRAM memory cells are especially vulnerable to NBTI since the input to one of the pMOS devices in the cell is always at logic “0.”In this paper, we proposed 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 bit line to vdd. So that NBTI problem is reduced. We show how fine-grained recovery boosting can be used to design the array of SRAM’s and evaluate their designs via Micro wind level simulations. By this we are also reduced power dissipation of the circuit. And also we designed 4bit SRAM array to improve the read and write ability of the circuit.

In future work, the plan to study the use of coarse-grained recovery boosting, which imposes less area overheads, for designing caches. Caches pose additional challenges, such as identifying when lines become valid to put them into the recovery boost mode. The plan to explore the use of techniques such as dead-block prediction 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] Bo, Z., et al. A Sub-200mV 6T SRAM in 0.13um CMOS. in Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International. 2007.

[2] Cheng, B., Roy, S., Roy, G., Brown, A., and Asenov, A. Impact of Random Dopant Fluctuation on Bulk CMOS 6-T SRAM Scaling. in Solid-State Device Research Conference, 2006. ESSDERC 2006. Proceeding of the 36th European. 2006.

[3] Jawar Singh, D.K.P., Simon Hollis, and Saraju P. Mohanty, A single ended 6T SRAM cell design for ultra-low-voltage applications. IEICE Electronics Express, 2008. 5(18): p. 750-755.

[4] Chang, L., et al., An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches. Solid-State Circuits, IEEE Journal of, 2008. 43(4): p. 956-963.

[5] Tae-Hyoung, K., et al. A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtual Ground Replica Scheme. in Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International. 2007.

[6] Wang, X., Roy, S., and Asenov, A., Impact of Strain on the Performance of high-k/metal replacement gate MOSFETs, in Proc. 10th Ultimate Integration on Silicon (ULIS 2009). 2009.

[7]J. Abella, X. Vera, and A. Gonzalez,(2007) “Penelope: The NBTI-aware processor”, in Proc. 40th IEEE/ACM Int. Symp. Microarchitecture.

[8]H. Akkary, R. Rajwar, and S. T. Srinivasan, “Checkpoint processing and recovery: Towards scalable large instruction window processors,” in Proc. Int. Symp. Microarchsitecture (MICRO), Dec. 2003, pp.423–434.

[9]P. Bose, J. Shin, and V. Zyuban, “Method for Extending Lifetime Reliability of Digital Logic Devices Through Removal of Aging Mechanisms,” U.S. Patent 7 489 161, Feb. 10, 2009.

[10]A. Cabe, Z. Qi, S. Wooters, T. Blalock, and M. Stan, “Small embeddable NBTI sensors (SENS) for tracking on-chip performance decay,”in Proc. Int. Symp. Quality Electron. Design (ISQED), Mar. 2009, pp.1–6.

[11]X. Fu, T. Li, and J. Fortes, “NBTI tolerant microarchitecture design in the presence of process variation,” in Proc. Int. Symp. Microarchitecture (MICRO), Nov. 2008, pp.399–410.