Logic-based Row Redundancy Technique Designed in 7nm FinFET Technology for Embedded SRAMs

Vivek Nautiyal, Nishant Nukala, Fakhruddin Ali Bohra, Sagar Dwivedi, Jitendra Dasani, Satinderjit Singh, Gaurav Singla and Martin Kinkade
Arm Inc, San Jose, California, USA, E-mail: vivek.nautiyal@arm.com

***Abstract*—In this paper, a row-redundancy circuit using latches is designed for 7nm FinFET ultra high density SRAM operating at 1.75 GHz. Input and faulty addresses are compared in parallel to the memory read access operation thus avoiding a major impact on access or address setup time. Latch output data is multiplexed with memory data and the impact on access time is only 7ps at SS/0.675V/-40°C corner. Data is written to redundant latches only when address comparison matches. The proposed circuit is implemented with no setup time impact and an overall area overhead of the proposed row redundancy scheme is less by 82% as compared to the area overhead of the conventional redundancy scheme.**

I. Introduction

It is a well-known fact that memories, especially embedded SRAMs have been the heart of modern System on Chip (SoC) designs. With shrinking geometries, the transistor count/density in memories has enormously increased to such an extent that the performance, power, yield and reliability of the overall die are greatly influenced by memory design. Errors of various kinds frequently creep into memories at smaller technology nodes, aggrevate especially at low supply voltages and hence significantly effect the chip yield. Mitigating these errors is easier with increasing the supply voltage, however this tremendously increases the power consumption making this solution inadequate and demanding for better approaches at both device and circuit levels.

Errors in SRAMs can be mostly classified as hard errors, soft errors, timing-voltage-margin related errors and performance related errors. Sources of these errors can be many. Hard errors are mainly due to the fabrication defects which can cause incorrect operation and damage the memory chip permanently. These include, single bit faults where one memory bit (SRAM bitcell) is faulty or row/column faults where a row or a column of the SRAM array is faulty. Hard errors increase rapidly with scaling where manufacturing processes and technologies become very sophisticated. Soft errors on the other hand, are due to noises or alpha particles incidents on the memory cells which result in information loss, but there is no physical damage to the devices. These can be repaired by rewriting the lost information. These errors also increase with device scaling and are significant in SRAM designs. Combination of redundancy techniques to repair the hard errors and Error Checking and Correction (ECC) techniques to repair soft errors have been shown in the literature to be very effective in reducing these errors [3]. The redundancy techniques proposed in the literature so far have a tremendous impact on SRAM timing and also the spare bitcells used are in general susceptible to various faults. The novel flip-flop (FF) based redundancy technique proposed in this paper addresses these issues and does not have any serious impact on the SRAM timing and is also not sensitive to faults as it is logic-based. Rest of the paper is organized as follows.

Section II discusses some of the previous work done in redundancy. Section III explains the conventional redundancy approach while section IV describes the novel flip-flop approach to row redundancy. Finally results are discussed in section V.

II. Prior Work Related to Redundancy

A few details regarding the previous work done on redundancy and which have been in the author's range of investigations have been provided in this section. Most of the previous work in this field is related to the different implementations of spare rows/columns (implementation details are similar to the flowchart shown in Fig1) and their modifications. Various replacement circuit techniques such as intrasubarray replacement, intersubarray replacement and subarray replacement have been described in [3]. Due to the increasing memory capacity, the memory array is divided heirarchically into sub arrays. Intrasubarray and intersubarray replacement techniques deal with replacing the faulty row/column in the same subarray and another subarray respectively. [8] proposes a new intrasubarray replacement scheme where a flexible relationship exists between spare lines and spare decoders. This novel proposal has been shown to have a better usage of spare lines and decoders and less probability of memory failure. Apart from intra and inter subarray techniques, subarray replacement techniques deal with DC characteristic faults, such as excessive stand-by current.

Traditionally, memory repairing involved replacement of one faulty column with one spare column. [9] shows how to use one spare column to repair multiple defective cells in multiple columns. This is done by selectively decoding the row address when generating the control signals for column MUXes. Experimental results have successfully shown higher probability repair rate compared to the traditional methods. Another excellent publication which discusses a novel redundancy approach is [11]. This paper proposes a redudancy technique where spare word/bit lines and spare decoders are provided on the chip to replace the faulty ones while still maintaining the same address. This approach showed a significant improvement in the number of usable bits on the wafer. Yamagato *et.al* in [12] describe a way to optimize the tradeoff between chip yield and area penalty. This "Distributed Globally Replaceable" scheme is proposed exclusively for row redundancy, whose main theme is as follows: If the number of defects exceed the number of redundant lines in one block

(bank), then redundant lines from other blocks are used to replace the defective word lines. This has shown only a 3% area increase (it is also shown that fewer redundant cells are sufficient) and 61% improvement in repair efficiency. The main aim of various redundancy approaches has always been to continually make an optimal tradeoff between the number of spare rows/columns in order to increase the yield and reduce area penalty [1]. This optimization was also dealt from an algorithmic perspective in a few publications like [5] and [4].

An important take away from all the redundancy approaches discussed above is: Row redundancy has been shown to have a biggest pay off since polysilicon design rules are more aggressive than metal/contact design rules. Circuit design trade offs have to be carefully made since row decoders are in the critical timing paths and new row redundancy techniques are to be developed to minimize timing impacts along with the aim of reducing the area penalty and increasing the yield [7]. [10] is a patent which exclusively discusses row redundancy while minimizing the impact on timing. However, the circuit complexity is huge as it uses pipeling and state machine approach. In this paper, we achieve similar performance as in [10] without much circuit complexity.

III. **Conventional Redundancy Scheme**

Redundancy technique is in general used to repair the hard errors. In the conventional redundancy approach, a spare row and/or column array of memory cells are provided along with the regular memory cells on the chip in advance. It is assumed that the faulty row/columns are identified and the faulty address bits are notified during wafer testing/Built-in Self Test (BIST). The regular address bits received for either the write or read operation are then compared with the already stored faulty address bits. If the comparison matches, a "match signal" is set high indicating the memory location which is faulty. The faulty row/column array is then disabled and operation happens from the redundant/spare row/column array. If the "match signal" is low (indicating no fault), the read/write operation happens from the regular array. Fig1 shows the flow chart of the conventional row/column redundancy approach and Fig2 pictorially illustrates the top level process of this scheme. Though figures1 and 2 depict the general principles of both row and column redundancy, the choice between them depends on the density of the SRAM array. If the density of the SRAM array is small then column redundancy is sufficient otherwise row redundancy along with column redundancy might become a necessity. Additionally, with the advent of EUV lithography due to intense scaling, defects are not only in the SRAM array, but also in the periphery, which further necessiates the need for row redundancy as all the issues cannot be addressed by column redundancy alone [2]. Conventional redundancy scheme works very well and had improved the yield significantly. However, there are several penalties to be considered, out of which the most important one is that, there is an intolerable impact on memory timing, specifically the address setup time (which inturn effects the cycle time), because of the entire process of comparison, disabling the faulty row/column and enabling the spare row/column [6].

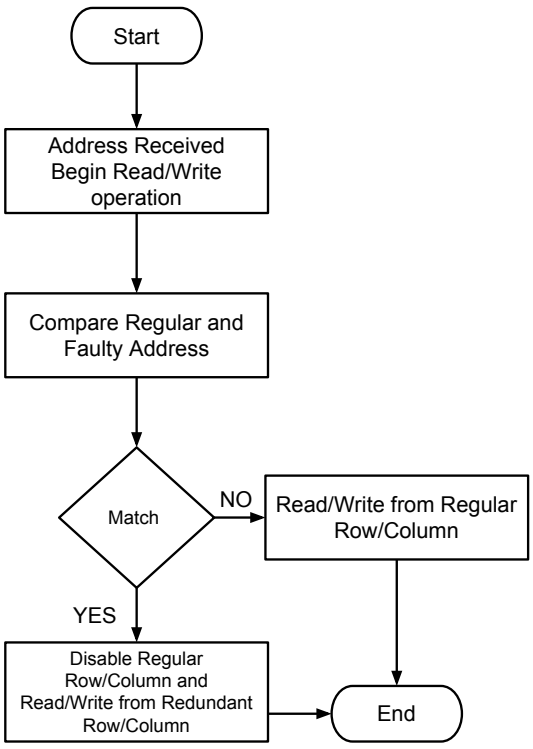

Fig. 1: Flow Chart for Conventional Row/Column Redundancy Scheme

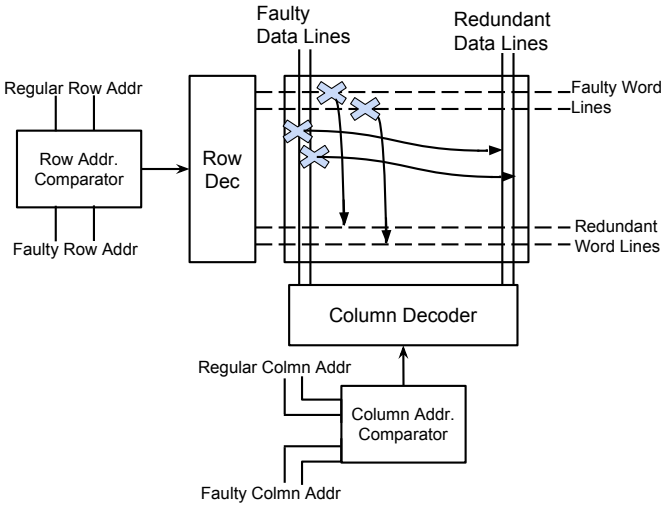

Fig. 2: Top-Level Block Diagram of Conventional Row/Column Redundancy Scheme

This also limits the maximum achievable frequency of the conventional redundancy scheme.

The flip-flop based row redundancy scheme described next, addresses this penalty and offers a solution to mitigate the impact on timing unlike the conventional approach and achieves a targetted frequency of 1.75GHz. It should be noted that, this redundancy technique is specifically row based. As there is no setup time penalty for column redundancy in both conventional and flip-flop based schemes, spare SRAM column has been used for column redundancy.

IV. **Flip-Flop Based Row Redundancy Scheme**

Fig3 shows the flow chart of the proposed flip-flop based row redundancy scheme. Unlike the conventional redundancy scheme which uses spare rows, this approach uses master-slave latches (flip-flops) to access the data. This simple, yet effective approach has no impact on the memory address setup timing when compared to the conventional approach. The input address bits received is compared to the faulty address while the data is simultaneously written or read (depending on the operation) into the current location based on the address received irrespective of whether the array is good or faulty. Note that the memory access time is greater than the comparison time, so a simultaneous address comparison and a memory read will not result in an incorrect read operation. Meanwhile, if there is a match between both the addresses, a "match signal" is set and depending on whether the operation is read or write, the data is either read from or written into the redundant latches provided. Since this approach uses an array of flip-flops

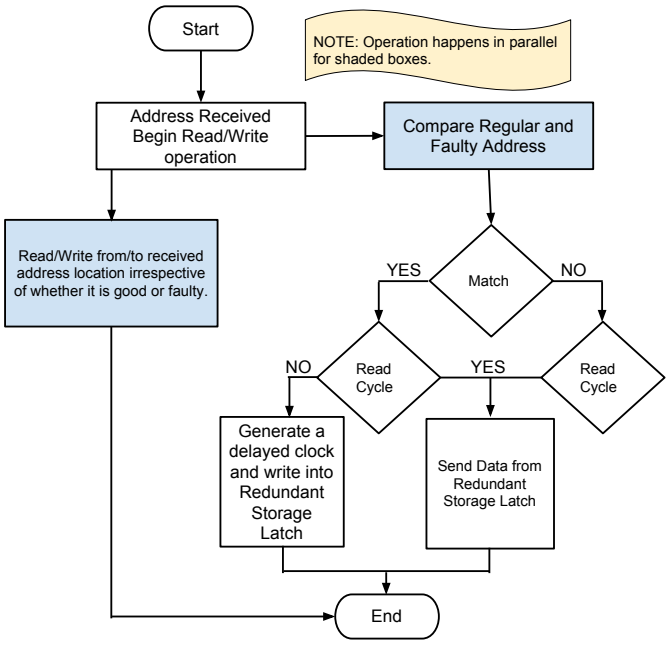

Fig. 3: Flow Chart for Flop Based Row Redundancy Scheme

and not any redundant memory array of rows, it can be used as a separate bolt-on mechanism as shown in the top-level block diagram in Fig4. Each FF block shown in Fig4 is a master-slave latch. If there is a match between the faulty address bits and the regular address bits and if the operation is a read, then the data is accessed out of the redundant "slave" latch in the FF block (blue line) otherwise the data from memory is read (red line). Q0 to QN/2 are the primary outputs of the MUXed redundant FF and memory signals. If the address bits match and the operation is a write, then the redundant latches are written. This approach has many advantages compared to the conventional row redundancy scheme and they are listed here.

- No setup time penalty
- No margin impact on the SRAM array
- Completely bolt-on
- Does not cost much area

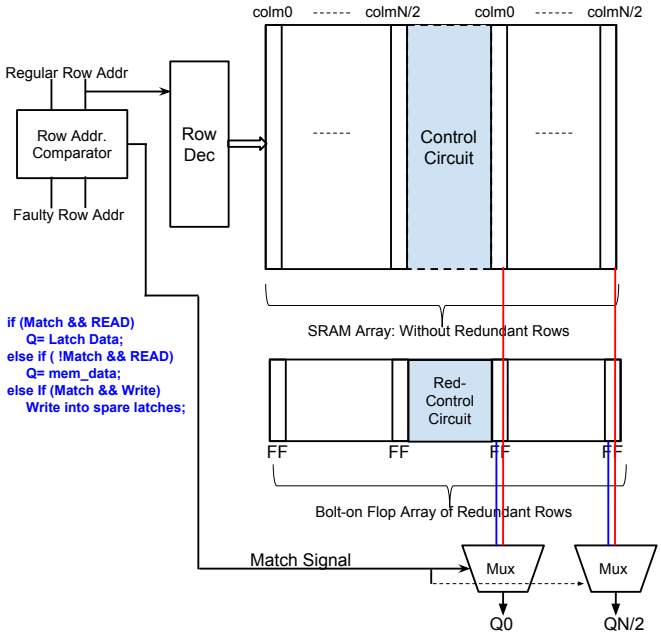

Fig. 4: Top-Level Block Diagram of Flop Based Row Redundancy Scheme

- Completely digital operation
- Built-in instance from the memory compiler and no additional work needed from the SoC side.

One caveat of this scheme is, for small instances area impact can be more, but row redundancy is preferred for large instances.

A. Operation and Signal Descriptions of Flop Based Redundancy Scheme

Depending on the requirement, the number of bolt-on redundant rows can be more than one as it is driven by yield and over all accumulated density of embedded SRAMs in SoC. However, based on the prevelant embedded SRAM accumulated density, generally one row and one column redundancy are sufficient for a 2MB SRAM. This section explains the signals needed and discusses the approach in more detail. Signals needed for column redundancy are omitted. TableI describes all the signals needed and Fig5 gives an in-depth view of the block diagram shown in Fig4. A brief description of the memory operation in Fig5 is as follows: An 8-bank memory array has been shown. The "Central Spine" block has all the control circuitry needed to operate the entire memory in a butterfly fashion. Two banks, for eg: bank0 and bank1 as shown in the figure share a local control circuitry called "Right shared Bank01/Left shared Bank01". This block includes the sense amplifiers, I/O's etc needed to access the data from these two banks. Output of each local "shared bank" block is MUXed (along with the redundancy block output - Q0_Red) and a "Banksel" signal helps in selecting the required final memory output. Note that the "Conv Redundant Rows" blocks are shown in the figure just to depict the floorplan of conventional redundancy scheme.

TABLE I: Signals needed for Flop-Based Redundancy Scheme

Signal Name	Description
D	Data Input
GWEN	Active Low, Write to the memory location when low else read from the memory location
WEN	Active Low, Bit Write Enable
WCLK	Fast clock to D master latch
RED_WCLK[7:0]	Clock signals for 8 slave latches. This is activated only when the "Match Signal" is triggered
iRED_WCLK[7:0]	Internal signal generated by interlocking WEN and RED_WCLK
Match Signal	Regular Row Address and Faulty Row Address comparison output
Q_MEM	Q output coming from the SRAM Array
Q_RED	Q output coming from the Redundant Slave latches
Q	Q output of the Slave latches Vs SRAM Array
RED_QSEL[7:0]	One of the RED_QSEL signals will go high and select the corresponding latch and transmit the data to the output
MEM_QSEL	Selects the final output either from the redundant latches or from the SRAM Array

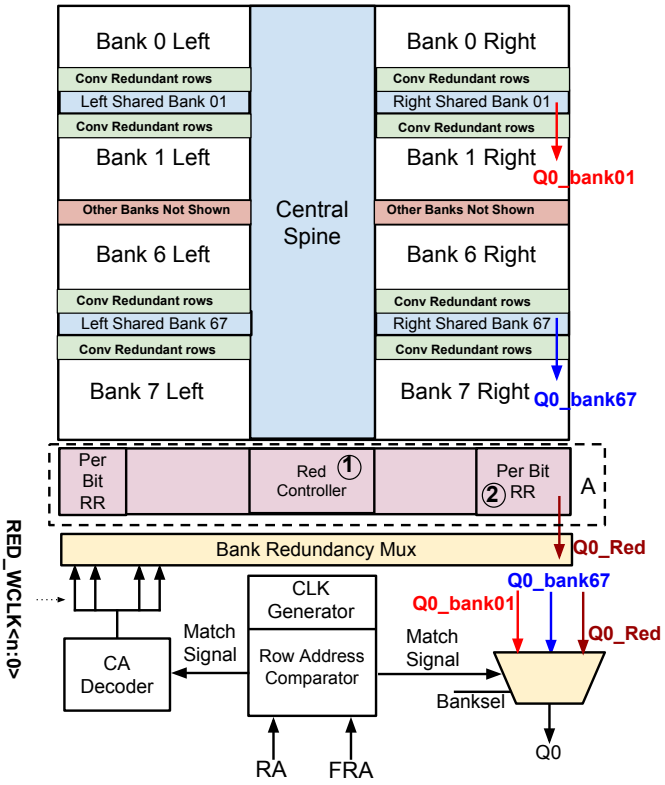

Fig. 5: Complete Block Level Schematic of Flop Based Row Redundancy Scheme

Redundant bolt-on block operation: The block shown as "A" in Fig5 is the bolt-on redundant flip-flop array. The redundancy controller block (circled number 1 in the Fig) and the per bit row redundancy (Per Bit RR) block (flip-flop block circled number 2) are enlarged in figures6 and 7 respectively. An inherent assumption made in this work is, the entire process of row redundancy happens only when the Row Redundancy Enable (RREN) signal is asserted. Referring to Fig6, when the Row Address (RA) and the Faulty Row Address (FRA) match in comparison, a "Match signal" is generated. This process of comparison happens parallelly to the memory write/read operation as explained in the flow chart (Fig3). The "Match Signal"and decoded Column Address (CA) generate a Redundancy Write Clock (RED_WCLK) which is used to write into the redundant latches. The "Match Signal" and GWEN generate a MEM_QSEL signal to select the final output either from the latches or the memory array. Other signals generated by the controller are a fast Write Clock (WCLK) and a fast RED_QSEL (generated using CA and GWEN) to select one of the outputs from the redundant slave latches. Solid block signals shown in the figure represent a bus and solid lines represent one bit signals.

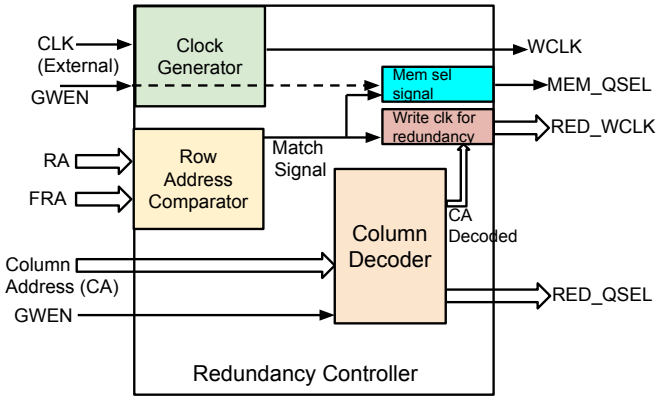

Fig. 6: Redundancy Controller in the Flop Based Redundancy Scheme

Each row redundancy bit, as already explained previously is a master-slave(s) latch combination. Fig7 depicts this configuration. When the operation begins, the master latch is loaded with the primary data (D) using an internally generated fast Write Clock (WCLK). A fast clock has been used here inorder to avoid large hold times on D input. Slave latches are operated only when the "match signal" is set. This is obvious as the iRED_CLK used in the slave latches depends on RED_WCLK which is inturn generated from the "Match Signal" (Fig6). All the experiments in this paper are done using mux=8 configuration. This means, there are one master latch and 8 slave latches (one for each column line in the memory) and an 8:1 mux to select the output data from one of these 8

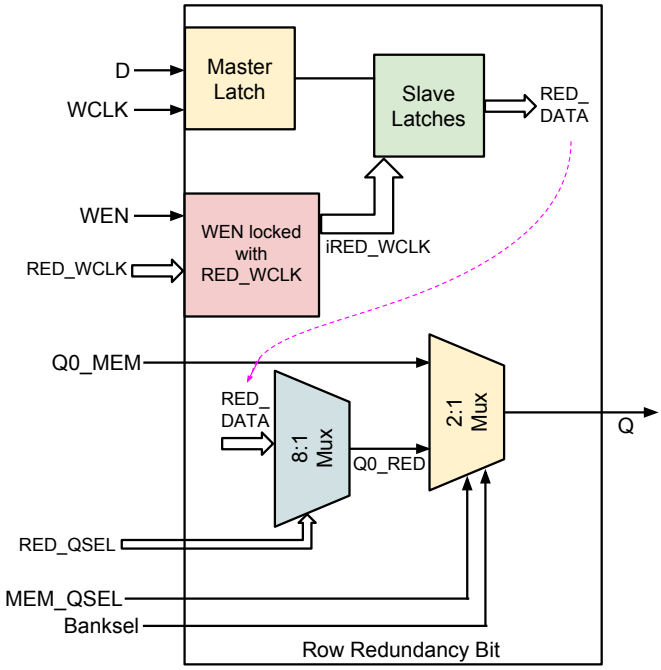

Fig. 7: Row Redundancy Bit in the Flop Based Redundancy Scheme

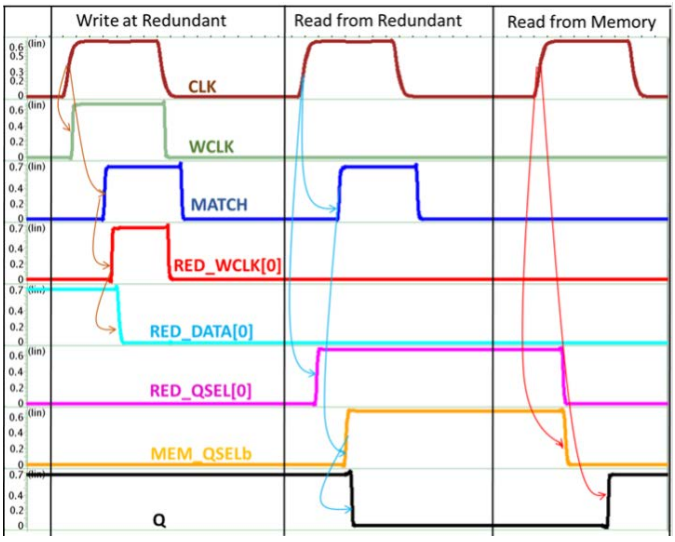

Fig. 8: Simulation Results for Flop Based Row Redundancy Scheme

slave latches in each row redundancy bit. Using such a master-slave(s) configuration saves a very significant amount of the chip area. The RED_WCLK signal is a 8-bit bus derived from CA and only one of its 8 bits is asserted at a time enabling only one of the 8 slave latches.

If the input data is not changing frequently, then the user can save power by not writing into the latches. This functionality is achieved by Write Enable (WEN) bit signal. If WEN is high, then writing into the latches is disabled. This feature is also incorporated here by interlocking the WEN and RED_WCLK signals and the newly generated iRED_WCLK is used as the clock for slave latches as shown in Fig7. Finally, MEM_QSEL signal is used to select between Q0_RED the output bit the of redundant slave latches and Q0_MEM the output bit of the memory array. Note that when MEM_QSEL goes high, Q0_RED is selected and the "Banksel" signal, shown in Fig5 is disabled thus having no performance or timing impact.

V. Results and Discussions

Fig8 shows the simulation results of the flop based redundancy approach. Three different phases of operation are shown. The first phase is writing into the slave latches when there is a match between the regular and faulty row addresses. The sequence of various signals shown is as follows: The first signal shown is the external clock signal (CLK). An internally generated fast write clock (WCLK) shown next is derived from CLK. This signal is used to write into the master latch in each row redundancy bit at the start of the operation. The comparator generates the "match signal" shown next which along with CA is used to derive the RED_WCLK bus. Only one bit signal RED_WCLK[0] shown in the waveform indicates that first latch has been written and the data written is RED_DATA[0]. This completes the write phase.

The second phase is the read from the redundant latches. Match signal is again generated and a RED_QSEL[0] signal derived from CLK, GWEN and CA is used as a select signal for the 8:1 mux as shown in Fig7. Finally, the MEM_QSEL (shown as MEM_QSELb in the figure) signal is triggered to select the output from the redundant latches. At this point, "Banksel" is disabled.

The last phase is the memory read incase there is no match between the addresses. MEM_QSELb signal goes low and the data is selected from the memory muxes.

Figures9 and 10 show the access and setup timings respectively for different row redundancy approaches at three Process-Voltage-Temperature (PVT) corners. A very large memory instance with 16384 words, 80 bits, mux 8 (80*8=640 columns) and 8 banks has been chosen and reported here for comparison. This instance is highly dense containing 1:1:1 bitcells (1 fin of pull-up PMOS, 1 fin of access NMOS and 1 fin of pull-down NMOS). All the simulation results shown are done using the commercially available tools in 7nm FinFET technology. Timings with no redundancy, conventional redundancy (comparator is internal to the memory) and proposed flop based redundancy schemes are shown in the figures. It can be seen that the increment in access time for the flop based redundancy is negligible (less than 1%) compared to the increment in setup time for conventional redundancy scheme. Also, at slow corner, since the circuit delay dominates (more than RC delay), the setup time for conventional scheme is more by 53% compared to proposed scheme (Fig10). If the overall access + setup time of the instance without redundancy, at slow/0.675V/-40°C corner is assumed to be Tps, then the conventional scheme has T+120ps and flop based scheme has T+7ps as their access + setup times. This clearly shows that the flop based redundancy has improved the overall timing overhead by 94.1% compared to the conventional scheme at

slow/0.675V/-40°C corner.

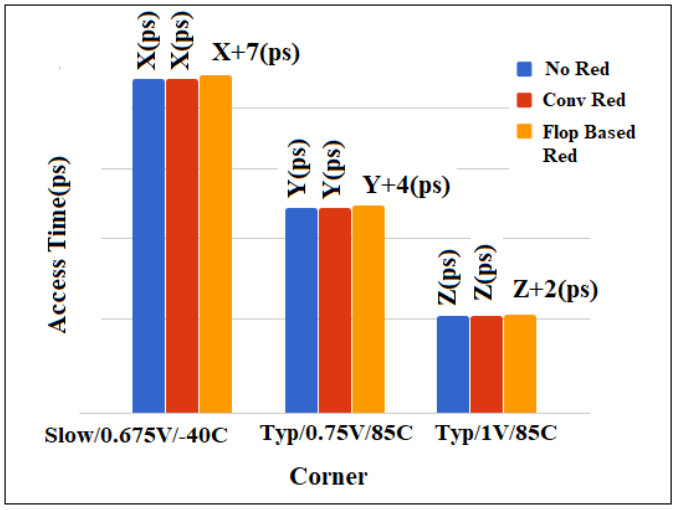

Fig. 9: Access times for different row redundancy schemes at various PVT corners

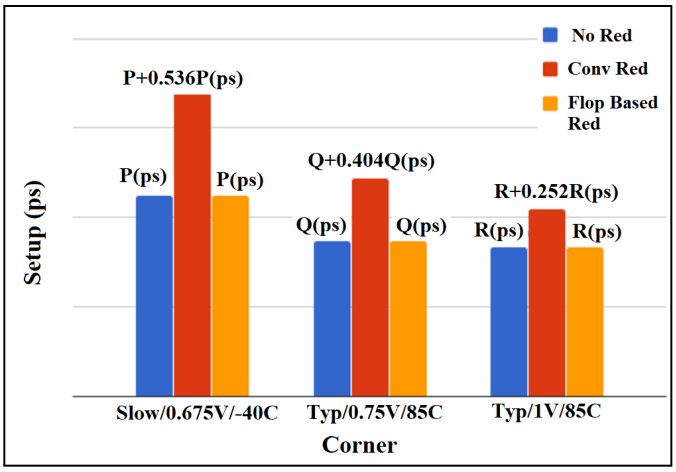

Fig. 10: Setup timings for different row redundancy schemes at various PVT corners

TableII shows the access time and the comparator delay of two instances: Biggest and thinnest at slow/0.675V/-40°C corner. It can be seen that the access time is much greater than the comparator delay and this justifies that flop based redundancy scheme will not have any impact on the setup time.

TABLE II: Access Time and Comparator delay for two different instances at slow/0.675V/-40°C corner

	Memory Access Time	Comparator Delay
16384x80m8fb8	Xps	(0.126*X)ps
16384x8m8fb8	Kps	(0.153*K)ps

Area Comparison: Conventional redundancy approach has 4 redundant rows in each bank (shown as "Conv Redundant Rows" in Fig5) which results in an additional area overhead of 1.2% compared to the biggest instance chosen (16384 words, 80 bits, mux 8 and banks 8) without any redundancy. Minimum of 4 redundant rows are needed in conventional scheme as this is due to the pitched layout limitation of the word line driver. The flop based redundancy scheme on the other hand has an area overhead of only 0.6% compared to the same instance without any redudancy. The area overhead for conventional scheme against no redundancy is 64 Contacted Poly Pitches (CPP's) and 35CPP's for the proposed scheme against no redundancy. This translates to about 82% extra area overhead in the conventional scheme compared to flop based scheme.

A Note on Power: In the proposed scheme redundant operation happens in parallel to the regular memory array access. However, since this is logic based, power consumption of redundant block is not significant compared to overall power. Also, it is physically close to bank multiplexer and IO pins with no additional routing overhead for input/output data. On this basis, it can be claimed that the overall dynamic power impact is less than 1%.

VI. Conclusions

In conclusion, a novel flip-flop based row redundancy scheme has been proposed and designed in 7nm FinFET technology, which helps in mitigating the impact on timing and has a minimum area overhead, unlike the conventional redundancy scheme.

References

[1] S. Eaton, D. Wooten, W. Slemmer, and J. Brady. A 100ns 64K Dynamic RAM using Redundancy Techniques. In *Digest of Technical Papers: 1981 IEEE International Solid-State Circuits Conference*, pages 84–85, Feb 1981.

[2] T. S. et. al. A 7nm FinFET SRAM Macro Using EUV Lithography for Peripheral Repair Analysis. In *Digest of Technical Papers: 2017 IEEE International Solid-State Circuits Conference*, pages 208–209, Feb 2017.

[3] M. Horiguchi and K. Itoh. *Nanoscale Memory Repair*. Springer, New York., 2011.

[4] C. T. Huang, C. F. Wu, J. F. Li, and C. W. Wu. Built-in Redundancy Analysis for Memory Yield Improvement. In *Proceedings. IEEE Transactions on Reliability, Vol-52, Issue.4*, pages 386–399, Dec 2003.

[5] W. K. Huang, Y. N. Shen, and F.Lombardi. New Approaches for the Repairs of Memories with Redundancy by Row/Column Deletion for Yield Enhancement. In *Proceedings. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol-9, Issue.3*, pages 323–328, Mar 1990.

[6] M. H. Kiyoo Itoh and H. Tanaka. *Ultra Low Voltage Nano-Scale Memories*. Springer, 2007.

[7] K.Kokkonen, P.Sharp, R. Albers, J.Dishaw, F.Louie, and R.Smith. Redundancy Technique for Fast Static RAMs. In *Digest of Technical Papers: 1981 IEEE International Solid-State Circuits Conference*, pages 1660–1671, Feb 1981.

[8] M. A. Masashi Horiguchi, Jun Etoh and K. Itoh. A Flexible Redundancy Technique for High-Density DRAM's. In *IEEE Journal of Solid State Circuits, Vol-26, No.1*, pages 12–17, Jan 1991.

[9] M. A. Masashi Horiguchi, Jun Etoh and K. Itoh. Improving Memory Repair by Selective Row Partitioning. In *Proceedings. 24th IEEE International Syposium on Defect and Fault Tolerance in VLSI Systems*, pages 211–219, 2009.

[10] Nautiyal. Memory Device and Method of Controlling Access to Such A Memory Device, US Patent: 7420859 B2, Sep 2008.

[11] S. E. Schuster. Multiple Word/Bit Line Redundancy for Semiconductor Memories. In *Proceedings. IEEE Journal of Solid State Circuits, Vol.SC-13, No.5*, pages 698–703, Oct 1978.

[12] T. Yamagata, H. Sato, K. Fujita, Y. Nishimura, and K. Anami. A Distributed Globally Replaceable Redundancy Scheme for Sub-Half-Micron ULSI Memories and Beyond. In *Proceedings. IEEE Journal of Solid State Circuits, Vol-31, No.2*, pages 195–201, Feb 1996.