SRAM Word-oriented Redundancy Methodology Using Built In Self-Repair

Jihyun Lee
Department of Electrical and Computer Engineering
Northeastern University
Boston MA, USA 02115
Email: jlee@ece.neu.edu

Young Jun Lee
Department of Electrical and Computer Engineering
Northeastern University
Boston MA, USA 02115
Email: yjlee@ece.neu.edu

Yong Bin Kim
Department of Electrical and Computer Engineering
Northeastern University
Boston MA, USA 02115
Email: ybk@ece.neu.edu

***Abstract*— In this paper, a word-oriented Built In Self-Repair (BISR) technique for SRAM bank is presented. All the repairs using BISR circuit are done during the reset period, and the process of referencing to the faulty line and its redundancy address are not required during the normal operation. The access time penalty is negligible and additional power consumption due to BISR circuit is kept minimum since it operates only during the reset time. Flexibility of this design technique allows to repair more numbers of faulty lines than conventional approaches with less area overhead.**

I. INTRODUCTION

Embedded memories gets higher in density as the technology moves toward deep submicron. Due to its high density, memory is more prone to faults. Even though chips are rejected in test, they still have high percentage of usable area. Those partial defects cause higher costs and lower yield. In order to circumvent this problem, many researchers proposed redundancy methodology replacing bad rows or columns with spare ones. Yield of embedded memories can be significantly enhanced using redundancy technique.

Conventionally, defective lines are replaced with redundant lines by switching between decoders [2], [1]. Access time penalty is inevitable in this method, which is not desirable for high-speed SRAMs. A parallel word access redundancy architecture was proposed in [3] with no access time penalty, but at expense of twice power consumption.

More recent approaches use BISR concepts [7], [5], [4], [6]. In [4], faulty address and its data are stored in the redundancy logic instead of using redundant rows or columns. A parallel access to the memory and the redundancy logic were proposed. This structure requires address comparison in BISR, causing extra power during the normal operation. In [6], BISR circuit requires fail address memory (consists of CAM) and a spare memory. This method also compares input addresses with stored addresses during normal operation increasing power consumption. Furthermore, it requires special memory CAM.

Another approach is to use soft fuse as a redirection device, employing interleaved shift register to program soft fuses. However, there is limitation on the fault line that can be repaired. Due to interleaved shift register, this scheme can fix only one bridge between rows, or a single defect in the odd address space and a single defect in the even address space. This method cannot repair two independent defects in the even(or odd) address space.

In this paper, BISR circuit which operates only during the chip reset is proposed. Therefore, no extra power due to BISR is consumed during normal operation. The proposed scheme uses redundancy rows, which does not require extra effort to design spare memory elements. Address comparison with faulty address is not required, because all the necessary repair is done during chip reset. Repair information is stored in the shift register cells, and the redirection of damaged line is designed such that access time penalty is negligible. This design technique repairs more number of faulty lines than conventional approaches by extending redundancy lines and BISR circuit with reasonable area overhead. There is no restriction on address space that can be repaired due to flexibility of its design scheme. The proposed method also improves external testability by providing serial input and output interface while minimizing pin count.

In Section II, word-oriented redundancy scheme with BISR concept is described. Repair information is generated from the Fuse Farm, and it is explained in Section III. Correct operation of BISR circuit is verified using verilog simulation in Section IV, followed by conclusion in Section V.

II. ROW REDUNDANCY SCHEME

The proposed row redundancy Built In Self-Repair (BISR) scheme is composed of three parts. First component is soft fuse, which is introduced as redirection devices in case defective word line is found. Second, shift register is employed to program soft fuses to skip defective row and use the next available word line instead. By implementing redirection to the next available word line address, wire interconnect delay is minimized. Therefore, this scheme reduces SRAM access time penalty significantly comparing to the case of redirecting to the redundant rows. Any desired number of repairs are possible with this methodology by simply expanding the number of redundant rows, soft fuses, and shift register. Third, keeper circuit is used to force the damaged word line to remain at V_{ss}. In the following subsections II-A and II-B, detailed methods to implement row redundancy scheme are presented.

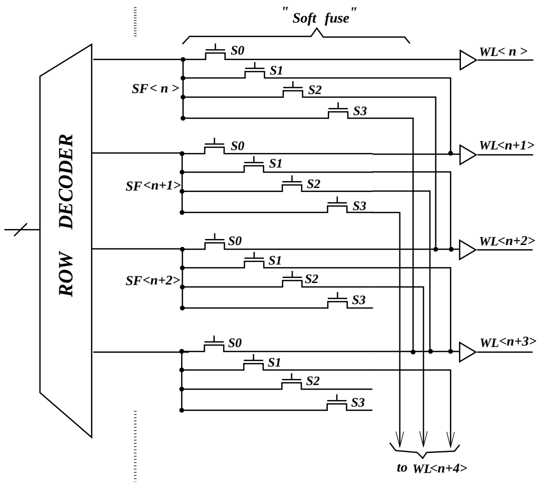


Fig. 1. Redirection method using Soft Fuse

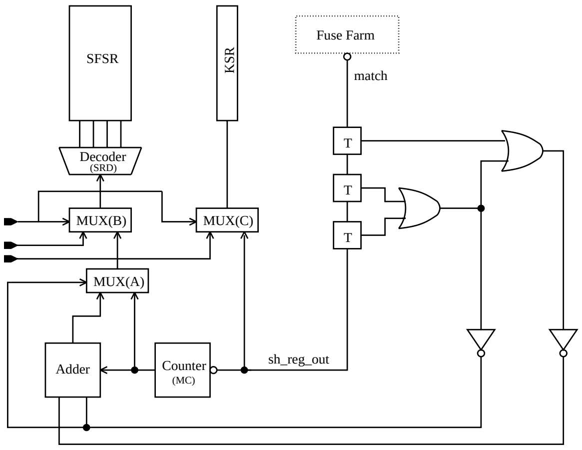


Fig. 2. Shift Registers and Peripheral Circuitry

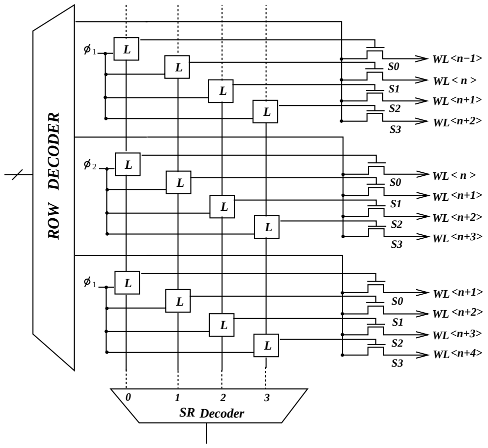


Fig. 3. Soft Fuse Shift Register (SFSR)

A. Soft Fuse

Figure 1 shows the soft fuse (SF) circuit as a redirection device. This figure demonstrates the case of four soft fuses per each word line (WL) that makes three repairs possible. In this example, three redundancy rows are needed. A NMOS transistor is used as a soft fuse, and each fuse redirect the address from the Row Decoder in the following way. If there is no defective row up to $WL_{<n>}$, soft fuse S_0 is on, and the other three S_1, S_2, S_3 are off. In this case, address from the row decoder and the topological address are identical. However, if there is a defect in $WL_{<n+1>}$, S_1 turns on while the other three fuses turn off in $SF_{<n+1>}$. S_1 skips the damaged row and redirects the address to the next word line $WL_{<n+2>}$. All the SFs must keep the same value as $SF_{<n+1>}$ until the next defective row is detected, shifting each WL by one topological address. In a similar way, if there is two defects (or three defects), S_2 (or S_3) is selected and shifts each WL by two (or three) topological addresses. Only one soft fuse in each SF is on, and control values of SF are programmed using shift register and its peripheral circuitry. By redirecting to the closest available address, access time penalty using redundancy is minimized.

B. Shift Register and Peripheral Circuit

Shift registers and its peripheral circuitry are shown in Figure 2. Shift registers are used to program soft fuses (denoted as $SFSR$) and to control keeper circuits for damaged rows (denoted as KSR). $SFSR$ consists of four columns of shift registers for repairing three damaged rows. Each cell (denoted as L) of the $SFSR$ shown in Figure 3 contains a binary control value for each soft fuse. KSR needs only one column of shift register whose cell is identical to that of $SFSR$.

The control values ($match$) comes from the bitstream generator in Fuse Farm explained in Section III. Low value of $match$ indicates damaged row, and high if no defect. The sequence of $match$ signal is programmed into KSR and deactivates corresponding damaged word line. However, peripheral circuitry is required to program $SFSR$ so that correct soft fuse turns on to redirect to the next available word line in case defect is present.

Match counter (MC) counts the number of defects from a stream of $match$ signal. Input of MC is active low, and increases its value by one whenever $match$ is low. Output of MC goes into the decoder logic (SRD) and selects the correct soft fuse among S_0, S_1, S_2, S_3. Consequently, any non-consecutive damages up to three can be handled. To account for two or three consecutive damaged lines, additional circuits such as MUX(A), Adder, transmission gate latches (T), OR gates, and Inverters are needed. Three Ts store stream of $match$ temporarily, and check if there is any consecutive defects (consecutive 0s). If consecutive defects are found, MUX(A) takes input from the Adder instead of the MC. First low signal increments MC and its output is added with additional number of consecutive low signal. MC keeps counting for all the following low signals so that SRD selects the correct SF when next high signal comes in.

Figure 4 shows the cell design for $SFSR$ and KSR with its clocking scheme. T also uses the same clocking method. Two

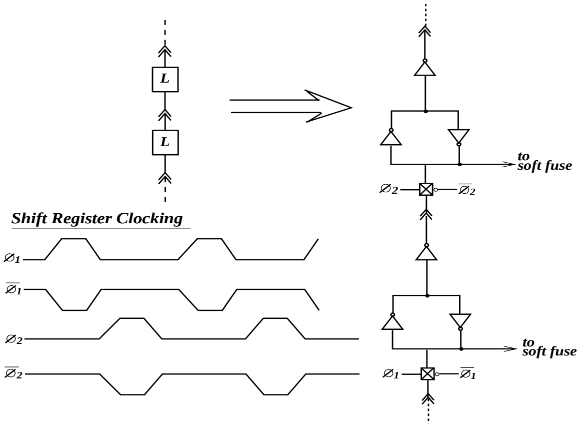


Fig. 4. Shift Register Cell (L) and Clocking Scheme

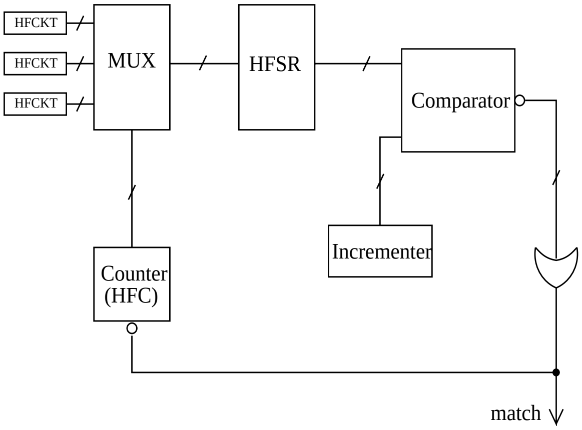
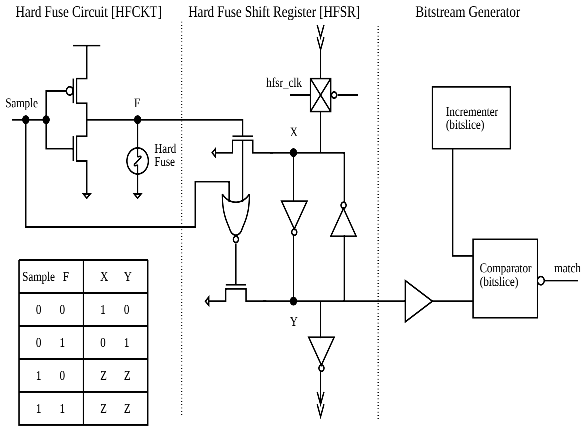


Fig. 5. Fuse Farm



Sample	F	X	Y
0	0	1	0
0	1	0	1
1	0	Z	Z
1	1	Z	Z

Fig. 6. Bit Slice of the Fuse Farm

non-overlapping clock (ϕ_1 and ϕ_2) must be used to prevent race problem. The shift registers do not need to run at high speed because programming of soft fuses and keepers are done during the chip reset. Therefore, special low frequency 2 Phase Clock Generator is used for safe operation of this Built In Self-Repair redundancy scheme. A $Test$ signal for the clock generator must be active during chip reset, and during the reset period only. Clocking must stop with $Done$ signal which becomes active after all the programming is done.

MUX(B) and MUX(C) provide external access to the shift registers $SFSR$ and KSR. This allows programming directly with serial test signals. Also, shift registers have an external interface output to allow testing of the fuse programming circuits by scanning out the programmed values.

III. FUSE FARM

Fuse farm generates bitstreams ($match$) which are used to program shift registers. Fuse farm consists of Hard Fuse circuits ($HFCKT$), Hard Fuse shift register ($HFSR$), and Bitstream Generator. To read in several defective word line addresses from HF, counter (HFC) and MUX are used as shown in Figure 5. When a $match$ is found (low signal) HFC choose next defective address in $HFCKT$ using MUX. Therefore, $HFCKT$ must be arranged in increasing order in address.

A. Hard Fuse Circuit and Hard Fuse Shift Register

Hard fuse circuit ($HFCKT$) consists of hard fuses that store the topological address of defective rows. Each hard fuses contain binary number by blowing the correct hard fuses with laser vaporization technique. For the example previously illustrated, up to three $HFCKT$ (three defective addresses) can be handled with three redundancy rows.

Figure 6 shows bit slice of the fuse farm. Low $sample$ signal allows address information from $HFCKT$ to be read in. Clock for the $HFSR$ (denoted as $hfsr\_clk$) must remain low in this case because $HFSR$ is for the external test purpose only. If $sample$ is high, X and Y node become high impedance state, which allows external test using $HFSR$. Serial input to the $HFSR$ is needed to save pin count. Clocking scheme explained in Section II-B also is applied to the $hfsr\_clk$.

B. Bitstream Generator

Bitstream Generator consists of Comparator, Incrementer, and OR gate. The incrementer runs through all the possible address values. The incrementer outputs are compared with the $HFCKT$ address. Address comparison is performed in bitslice as shown in Figure 6. If the bit address information is matched, comparator output is low (high if not matched). Outputs of bitsliced comparator are combined through the OR gate and produce $match$ signal. $Match$ is low if damaged address is found, and is high otherwise. Streams of $match$ signal are repair information to program $SFSR$ and KSR.

IV. SIMULATION RESULT

Figure 7 shows the BISR circuit interface with SRAM bank. Functionality of BISR circuit is verified using Verliog simulation, and circuit is verified using HSPICE with 0.18 μm CMOS technology using 3V power supply. Test condition for the worst case process/design corner is 2.7V at 110°C.

Figure 8 shows the simulation results of shift registers and peripheral circuitry. The *sh_reg_out* signal is 3 clock delayed

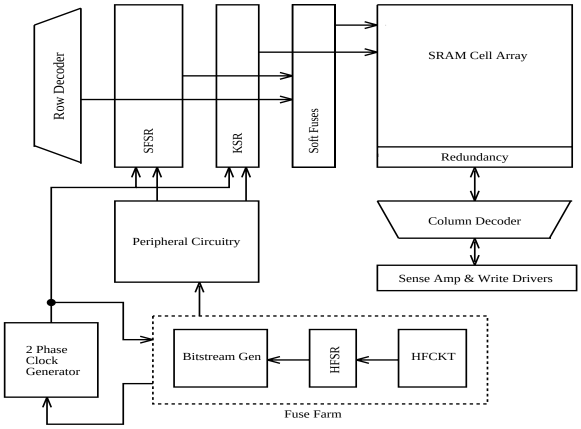


Fig. 7. BSIR interface with SRAM bank

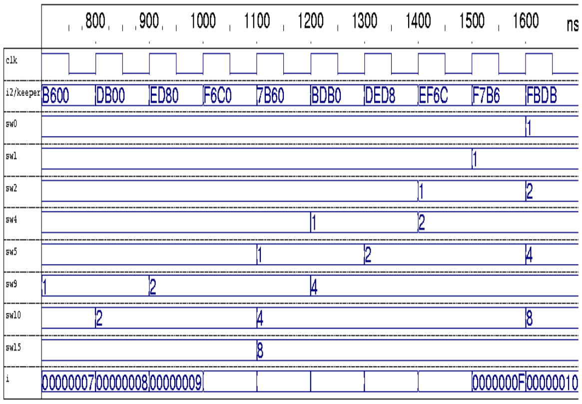


Fig. 8. Simulation Results of Shifter Register and Peripheral Circuitry

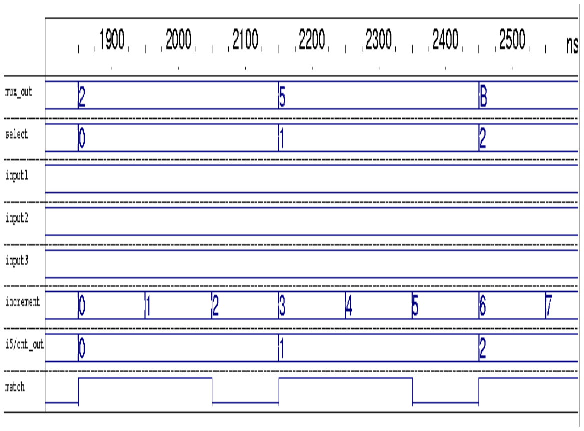


Fig. 9. Simulation Results of Fuse Farm Circuit

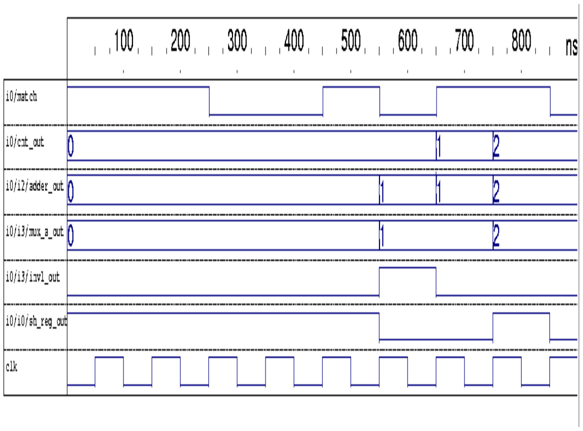


Fig. 10. Simulation Results of BSIR interface with SRAM bank

compared to *match* signal and the output of mux(A) is changed by *sh_reg_out*. The output signals for fuse farm are shown in Figure 9. When the address of defected line is matched with the output of incrementer, the *sh_reg_out* signal is generated.

The simulation results of output signals for BISR circuit is shown in Figure 10. For this simulation, the defect lines are assumed to be 3_{rd}, 6_{th}, 11_{th} bits and the number of wordline is assumed to be 16. Even though it is not realistic, it is enough to explain the main idea of this paper. From Figure 10, the output signals of keeper transistors are $FBDB$, and the output signals for soft fuse are exactly coincided with that of keeper.

V. CONCLUSIONS

A word-oriented BISR technique for SRAM bank is introduced. Advantages of the proposed techniques are: (1) low power consumption of BISR (2) negligible access time penalty (3) improvement in flexibility and testability (4) minimum area overhead (5) yield enhancement.

REFERENCES

[1] Stanley Schuster, *"Multiple Word/Bit Line Redundancy for Semiconductor Memories"*, IEEE Journal of Solid State Circuits, Vol.SC-13, No.5, October 1978, pp.698-703.

[2] T. Mano and M. Wada and N. Ieda, M. Tanimoto, *"A Redundancy Circuit for a Fault-Tolerant 256K MOS RAM"*, IEEE Journal of Solid State Circuits, Vol.SC-17, No.4, August 1982, pp.726-731.

[3] K. Sasaki and K. Ishibashi and T. Yamanaka and N. Hashimoto and et. al, *"A 9-ns 1-Mbit CMOS SRAM"*, IEEE Journal of Solid State Circuits, Vol.24, No.5, October 1989, pp.1219-1225.

[4] V. Schober and S. Paul and O. Picot, *"Memory Built-In Self-Repair using redundant word"*, IEEE International Test Conference, 2001, pp.995-1001.

[5] I. Kim and Y. Zorian and G. Komoriya and H. Pham and F. Higgins and J. Lewandowski, *"Built In Self Repair For Embedded high density SRAM"*, IEEE International Test Conference, 1998, pp.1112-1119.

[6] A. Tanabe and T. Takeshima and H. Koike and Y. Amimoto and M. Takada and et. al, *"A 30-ns 64-Mb DRAM with Built-in Self-Test and Self-Repair Function"*, IEEE Journal of Solid State Circuits, Vol.27, No.11, November 1992, pp.1525-1533.

[7] T. Kawagoe and J. Ohtani and M. Niiro and T. Ooisih and M. Hamada and H. Hidaka, *"A Built-In Self-Repair Analyzer (CRESTA) for Embedded DRAMs"*, IEEE International Test Conference, 2000, pp.567-574.

[8] L. Youngs and S. Paramanandam, *"Mapping and Repairing Embedded-Memory Defects"*, IEEE Design and Test of Computers, January 1997, pp.18-24.