2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems

Improving Memory Repair by Selective Row Partitioning

Muhammad Tauseef Rab, Asad Amin Bawa, and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
Email: {mrab, bawa, touba}@ece.utexas.edu

Abstract

A new methodology for improving memory repair is presented which can be applied in either manufacture time repair or built-in self-repair (BISR) scenarios. In traditional memory repair, one spare column can only replace one column containing a defective cell. However, the proposed method allows a single spare column to be used to repair multiple defective cells in multiple columns. This is done by selectively decoding the row address bits when generating the control signals for the column MUXes. This logically segments the spare column allowing it to replace different columns in different partitions of the row address space. The hardware is the same for all chips, but fuses are used to customize the row decoding circuitry on a chip-by-chip basis. An algorithm is described for choosing which row address bits to decode given the defect map for a particular chip. This additional degree of freedom allows customization based on the defect map of a chip and increases the effectiveness of the proposed scheme in comparison to traditional memory repair. Experimental results show that, when compared with traditional schemes of similar complexity, the proposed scheme achieves a higher probability of repairing defects.

1. Introduction

Yield loss in integrated circuits (ICs) due to SRAM-based memory failures are expected to increase as transistor sizes shrink to smaller dimensions and voltage levels are reduced. Caches constitute in excess of 50% of modern SOC and processors' area and account for more than 80% of the transistor count [Roelke 07]. In addition to the fact that the SRAM cell is the most frequently used cell, it also uses the smallest geometry transistors to increase area utilization which makes it more susceptible to both device electrical and geometrical variation [Mukhopadhyay 05]. To reduce yield loss due to defects in memory cells, the conventional approach is to add spare rows and columns to a memory [Schuster 78], [Zorian 03]. A test procedure is used to generate a defect map which indicates which cells in the memory are defective. Based on the defect map, the memory is reconfigured to use the spare rows and columns to bypass defective cells [Kuo 87], [Wey 87], [Hemmady 89]. If it is not possible to bypass all the defective cells, then the repair process fails. The memory reconfiguration can be done either at manufacture time with fuses, or it can be done with a built-in self-repair (BISR) scheme [Kim 98].

Fig. 1 shows a small example of a 4x4 memory with a spare row and spare column. Note that a spare column can only be used to repair a single defective memory cell unless there are multiple defective memory cells in the same column. This paper proposes a way to enhance the repair capability of a memory with a given number of spare columns. It allows a single spare column to be used to repair multiple defective cells in different columns. This is done by

1550-5774/09 $26.00 © 2009 IEEE
DOI 10.1109/DFT.2009.20

IEEE computer society

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

decoding a small number of row address bits when determining which column will be replaced by the spare column. Effectively, this logically segments the spare column and increases the number of defective cells that it can repair. The row address bits that are decoded are selected based on the defect map for the memory. Given the defect map, an algorithm is described for selecting the best set of row address bits to decode in order to repair as many defective cells as possible. The additional degree of freedom in selecting which row address bits to decode on a chip by chip basis greatly enhances the efficiency of the proposed method and increases the probability of being able to repair a memory and thereby improves overall yield. The cost for the proposed method is some additional fuses and logic to select which row address bits to decode.

The proposed method can be used for manufacture time reconfiguration as well as for BISR schemes. The proposed method provides a way to deal with rapidly increasing defective bit rates as geometries are scaled down and voltage levels are reduced. It also has applications in future nanoelectric technologies where defect rates are expected to be very high.

This paper is organized as follows. In Sec. 2, an overview of the proposed scheme and how the memory segmenting is performed is given. In Sec. 3, the repair algorithm is explained. Experimental results are presented in Sec. 4, and Sec. 5 is a conclusion.

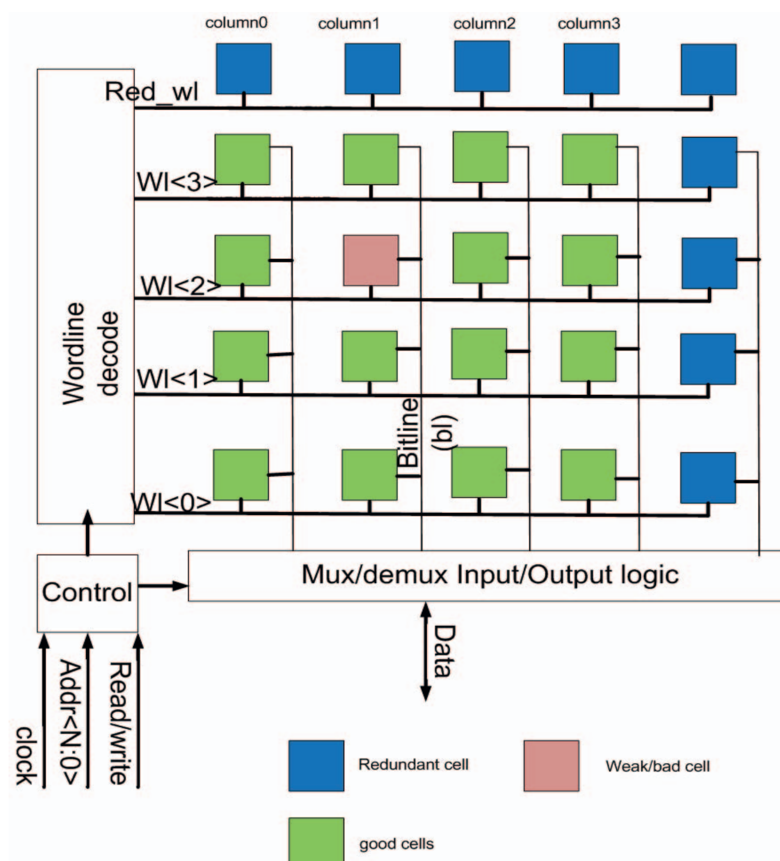


Figure 1. 4 Rows x 4 Columns Memory Block

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

2. Overview of Proposed Scheme

The main motivation behind the proposed scheme is the inherent inefficiency of the conventional approach. Typically an additional spare column is used in its entirety when repairing a defective cell. This implies that in order to repair multiple defects, the SRAM memory should typically have as many number of spares available as there are defects in the memory. This is generally true unless a single column in the memory has more than one defect; in which case the spare, when swapped out with the faulty column, ends up repairing both of the defects. Such scenarios are rare, thus multiple spares are added to SRAMs when more defects need to be tolerated. This linear dependency will lead to excessive overhead as SRAMs continue to be scaled and have increasingly higher defect rates. The proposed approach increases the effectiveness of spare columns by allowing a single spare column repair defects in multiple columns by partially decoding the row address when determining which column is replaced by the spare.

The proposed approach makes use of existing chip repair methodologies. The new aspect is given the defect map, the proposed approach selects *n* row address bits to decode so as to allow all defective cells in the memory to be bypassed. Moreover, the proposed scheme requires the ability to store the information on-chip as to which row addresses to decode and which column to replace in each partition of the row address space. This is illustrated in the high level block diagram of the proposed scheme given in Fig. 2.

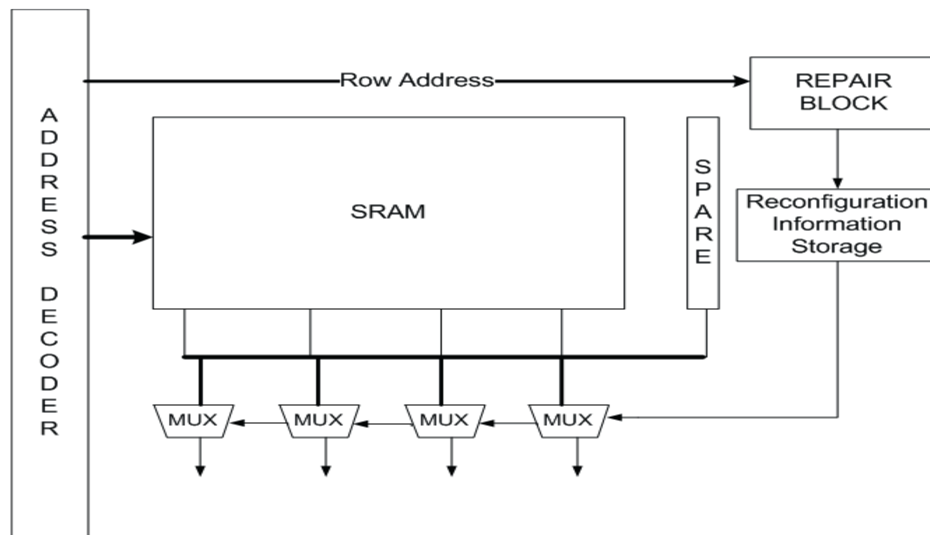


Figure 2. Block Diagram

3. Repair Algorithm

The memory is initially designed with a certain number of spare columns (maybe one, but can be more than one) and the ability to decode a certain number of row address bits (*n*). This hardware is fixed for all chips. Given the defect map for a particular chip, the proposed repair algorithm is used to configure the hardware to bypass the defective cells. The algorithm first selects the *n* row address bits to decode based on the defect map and the number of spare columns available. Decoding the *n* row address bits effectively partitions the row address

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

space into 2^n different partitions. The spare column can be used to replace one column in each of the 2^n different partitions. So the goal is to do the partitioning in such a way as to have only a single column in each partition having a defect. So if there are multiple defects in different columns, the partitioning should be done such that each partition contains only one defect. This is done by representing each defect's row address as a binary number, and then forming a matrix of the XORs of all the pairwise row address combinations. This is illustrated in Figure 3 where the row addresses of four defects are *{000, 010, 100, 111}*. Each of the 6 rows in the table correspond to one of the 6 pairwise combinations of the row addresses of the defects, and each column corresponds to a row address bit. An entry in the matrix is a '1' if the pair of row addresses of the defects differ in that bit, and a '0' if they are the same. For example, the row addresses for the defect at "000" and at "010" differ in bit 1, but not in bits 0 and 2. The goal is to select *n* row address bits to decode such that each defect is in a different partition. This corresponds to finding a column cover for the matrix in which a set of columns is selected such that it contains at least one '1' in each row. A column cover of this matrix ensures that the selected row address bits will distinguish every defective row such that no partition will contain more than one defect. If it is not possible to find a column cover with *n* or fewer columns, then the memory is irreparable. Many efficient column covering algorithms exist [Beasley 87].

Once the *n* row address bits have been selected, The fuses are programmed to hardcode which row address bits to decode as well as the column that is replaced by the spare in each partition. This can be done by implementing a small ROM or fuse box. The ROM drives the select lines of the column muxes which do the actual swapping of the faulty cell(s) with the good cell(s) from the spare element.

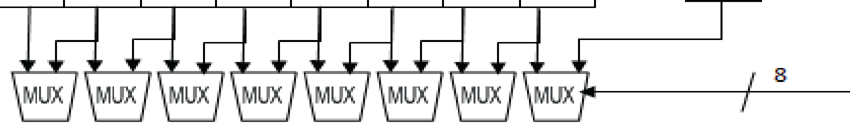
The repair procedure is now illustrated with a small example. Assume that $n=2$, so two row address bits are used to partition the spare column in a manner that it can map each defect in a unique partition of the spare column. The memory used in this example has 8 rows and 8 columns (C). The row addresses for the defects are *{D0, D1, D2, D3} -> {000, 010, 100, 111}*. The XOR between each pair of the row addresses with defects is performed and stored in a matrix as shown in Fig. 3. A column covering procedure is then performed which identifies a two bit column cover using bit 2 and bit 1. Those two row address bits are connected (with fuses) to a ROM (fuse box). The contents of the ROM are set so that columns 6, 4, 2, 1 are bypassed when the row address bits are 00, 01, 10, 11, respectively. The output of the ROM goes to a decoder which decodes the column to bypass and generates the appropriate set of MUX select signals. For example, if column 2 is to be bypassed, then the MUX select signals for leftmost 3 MUXes are set to select their right input, and the rest of the MUX select signals select their left input. This uses the spare column and skips column 2. The additional circuitry required to implement the proposed scheme for the above example is illustrated in Fig. 3. The fuse box is 2^n (n=2) by $\log_2 C$ (C=8) wide. Additionally a $\log_2 C$ to C (3 to 8) decoder is needed to generate the C (8) bit MUX select signals.

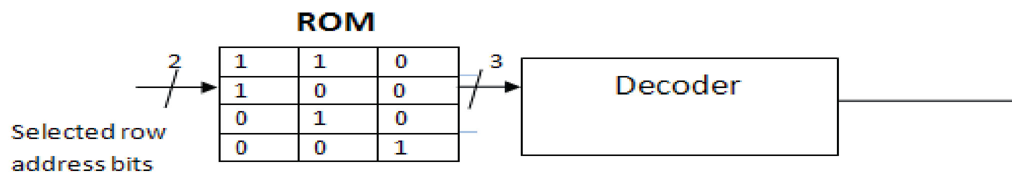
Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

Row Address #	Bit 2	Bit 1	Bit 0
(000 ⊕ 010)	0	1	0
(000 ⊕ 100)	1	0	0
(000 ⊕ 111)	1	1	1
(010 ⊕ 100)	1	1	0
(010 ⊕ 111)	1	0	1
(100 ⊕ 111)	0	1	1

8 x 8 Memory block

C7	C6	C5	C4	C3	C2	C1	C0	Spare Column
1	X	1	1	0	0	1	1	0
1	1	0	0	0	0	0	1	0
0	0	0	X	1	1	0	1	1
0	0	0	1	1	0	1	0	1
0	0	1	0	1	X	0	1	1
1	1	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	0
1	0	0	0	0	0	X	1	1





1	1	0
1	0	0
0	1	0
0	0	1

Figure 3. XOR Table and Example

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

4. Experimental Results

Several simulations were performed to evaluate yield improvements using the proposed selective row partitioning. For the first set of simulations, two, three and four random defects were injected across different memory block sizes and compared the traditional probability of repair with a single spare column versus the proposed scheme using a single spare column and two row address bits for decoding. Fig. 4, 5 & 6 show the results of these simulations. These simulations were mainly done to see the effectiveness of the proposed scheme in fixing multiple defects, and also to investigate if the size of the memory had any impact on the effectiveness of the proposed scheme.

As clearly seen from the results shown in Fig. 5 and Fig. 6 the proposed scheme is close to 100% effective for two and three defects and there is almost no dependency on the memory block size. However for four defects, shown in Fig. 6, the effectiveness of our approach is clearly dependent on the number of rows. This result is understandable and expected since the more rows in a block, the more likely it is to find two row bits to partition the memory with one defect in each partition. Note that for the case of a traditional single spare, the chance of repairing multiple defects is very low as it depends on the defects all being in the same column. The probability of this decreases as there are either more columns or more defects.

Fig. 7 summarizes the comparison between the traditional and proposed scheme for a 9-row-address-bit and 32-column-bits memory block using one spare column and 2-address bits for decoding (*n=2*). As can be seen below the proposed scheme is close to 100% effective for two or three defects, the reason it is not a 100% effective is due to the small chance of defects in the same row. Such defects can obviously not be distinguished using row address bits for decoding.

For four defects the proposed scheme still achieves close to 80% probability of repair. This reduction in the probability of repair going from three to four defects is also expected since partitioning the memory using 2-row-address-bits requires that each of the four partitions have exactly one defect and there are cases where a solution does not exist.

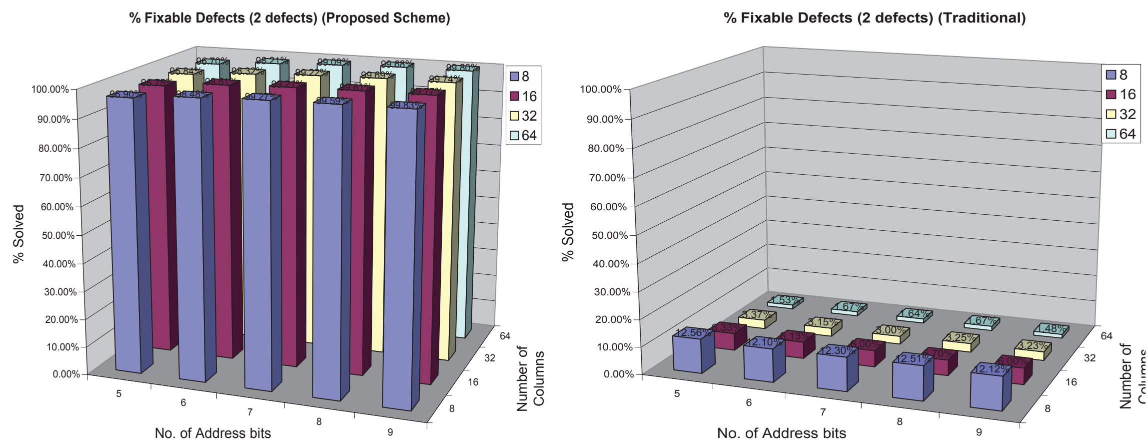


Figure 4. Repairable Defects across Different Memory Block Sizes (2 Defects)

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

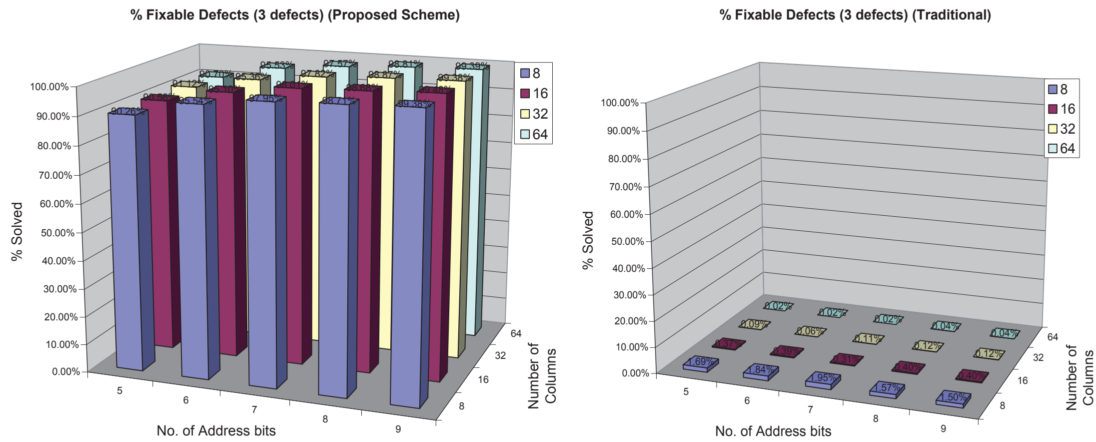


Figure 5. Repairable Defects across Different Memory Block Sizes (3 Defects)

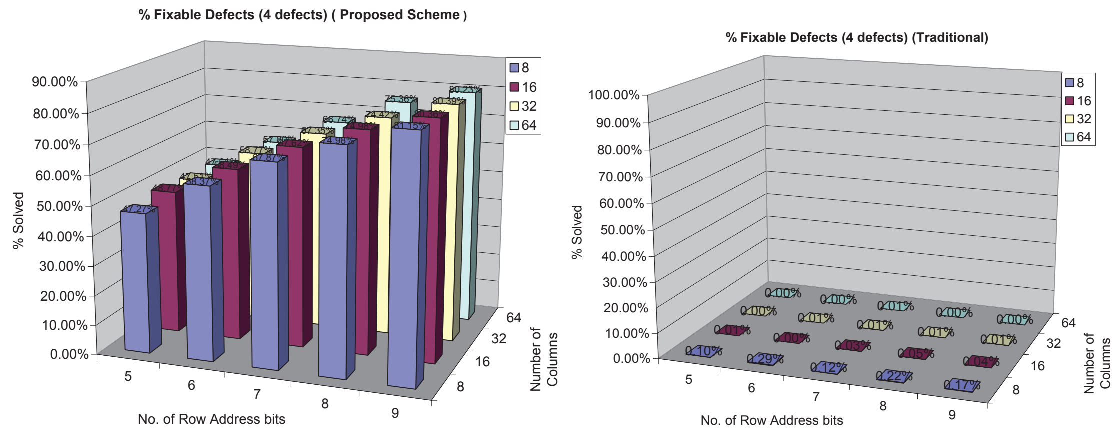


Figure 6. Repairable Defects across Different Memory Block Sizes (4 Defects)

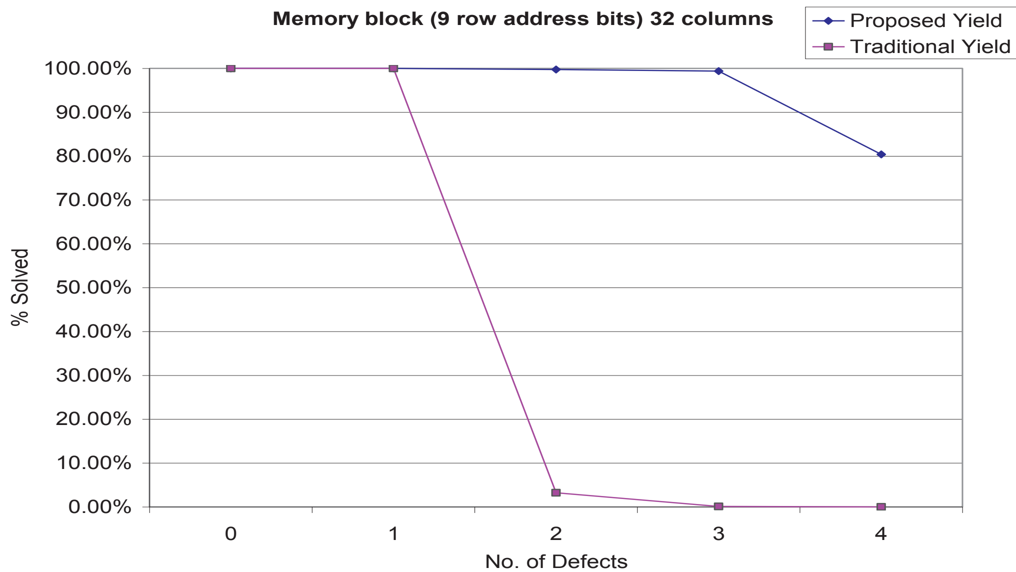


Figure 7. Repairable Defects Comparison Summary Decoding 2 Row Bits

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

Fig. 8 shows similar results Fig. 7 except in this case 2-spare columns are being used and an attempt is made to solve up to eight defects. This was done to show the scalability of the proposed scheme. Note that in this case, the same row address bit decoder is used for both spares. Alternatively, one could use separate row address bit decoders for each spare, and in that case the effectiveness of each spare would be equivalent to what was shown in Figure 7.

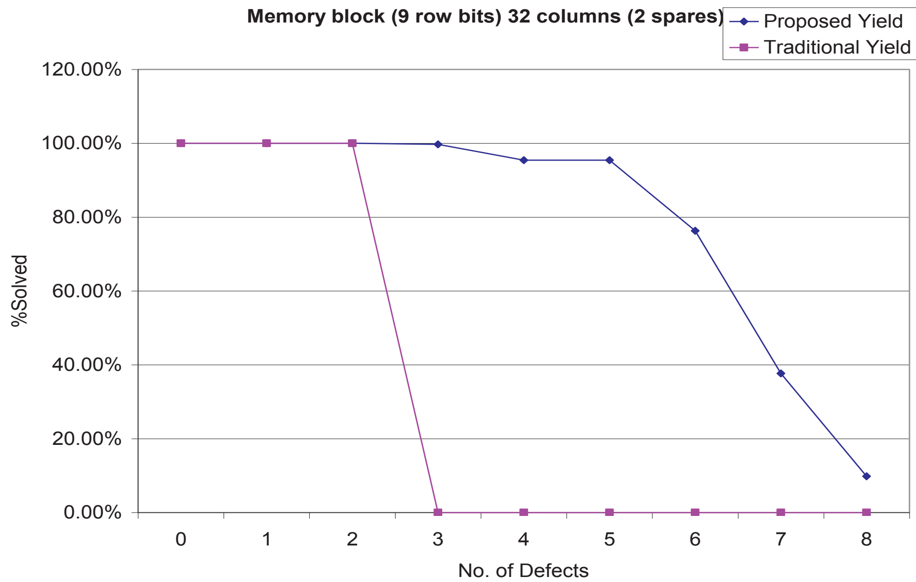


Figure 8. Repairable Defects Comparison Summary using 2 Spare Columns

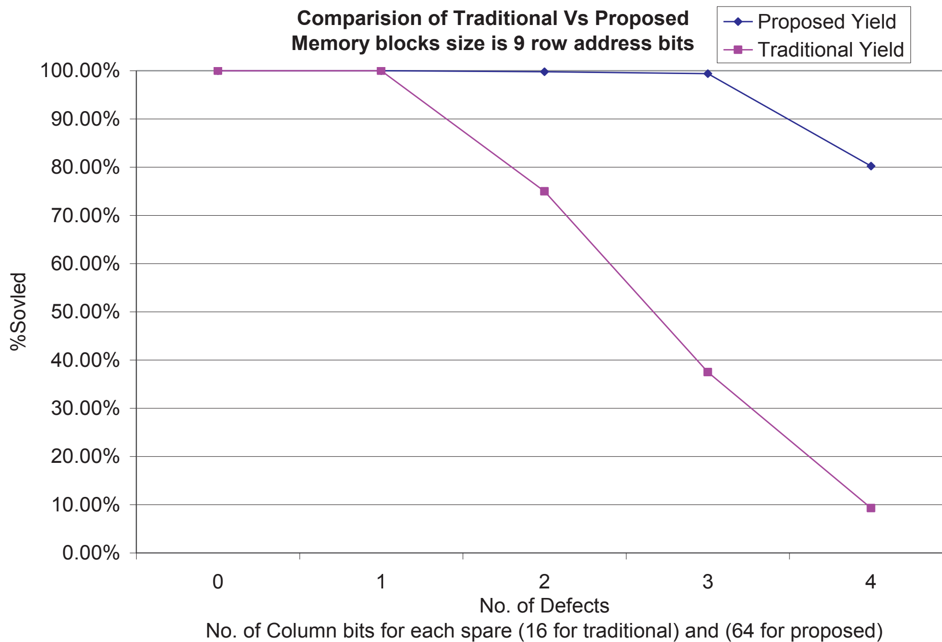


Figure 9. Repairable Defects using 2-Row-Bit Decoding with Wider Memory Block for Proposed Scheme

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.

The first set of experiments, summarized in Fig. 7 clearly showed the effectiveness of the proposed scheme in fixing multiple defects, however there is an area overhead cost associated with implementing the proposed scheme since it requires extra logic to implement (fuses, muxes, etc). An alternative to the proposed approach for improving repair capability would be to simply use the traditional approach with more spares. To compare these two alternatives, experiments were performed considering four memory blocks having 9-row-address-bits and 16-column-bits in which the traditional approach has one spare for each block. For the proposed scheme we assumed a single memory block with 9-row-address-bits, 64-column-bits and one spare column. The results are shown in Fig. 9. The probability of fixing defects with the proposed scheme is clearly better with less spare columns. The results for the traditional approach below can also be probabilistically calculated and confirm the numbers obtained through simulations. In this particular example, it has been shown that by reducing four spare columns down to one with the proposed approach, the probability of repairing defects is better. This advantage comes from the ability to select which row address bits to decode on a chip-by-chip basis such that the hardware is effectively customized to the defect map for a particular chip.

5. Conclusions

In this paper, a scheme was presented for selectively decoding row address bits to repair multiple defects using a single spare column. This approach is a logical extension to current approaches where a single spare column replaces another column that has a defect. The proposed scheme requires some additional logic and fuses to implement, however it was shown that when comparing the proposed scheme to the traditional approach of increasing the number of spare to tolerate more defects, for similar complexity, the proposed scheme achieves a higher probability of repairing defects (Fig. 9). The optimal number of row address bits to decode and memory block size to use would depend on the process, architecture, and defect rate.

References

[Beasley 87] J.E. Beasley, "An Algorithm for Set Covering Problem," *European Journal of Operational Research*, pp. 85-93, Jul. 1987.

[Kim 98] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F.P. Higgins, and J.L. Lewandowski, "Built In Self Repair for Embedded High Density SRAM," *Proc. of International Test Conference*, pp. 1112-1119, 1998.

[Kuo 87] S.-Y. Kuo and K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," IEEE Design & Test, Vol. 4, Iss. 1, pp. 24-31, Feb. 1987.

[Mukhopadhyay 05] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS," *IEEE Trans. on Computer-Aided Design*, pp. 1859-1880, Dec. 2005.

[Hemmady 89] V. Hemmady and S.M. Reddy, "On Repair of Redundant RAMs", *Proc. of Design Automation Conference*, pp. 710-713, June 1989.

[Roelke 07] G.R. Roelke, R.O. Baldwin, "A Cache Architecture for the Extremely Unreliable Nanotechnologies," *IEEE Trans. on Reliability*, Vol. 56, Jun. 2007.

[Schuster 78] S.E. Shuster, "Multiple word/bit line redundancy for semiconductor memories," *IEEE Journal of Solid-State Circuits*, Vol. SC-13, pp. 698-703, 1978.

[Wey 87] C.-L. Wey and F. Lombardi, "On the Repair of Redundant RAM's", *IEEE Trans. on Computer-Aided Design*, Vol. 6, No. 2, Mar. 1987.

[Zorian 03] Y. Zorian, and S. Skoukourian, "Embedded-Memory Test and Repair: Infrastructure IP for SOC Yield," *IEEE Design & Test of Computers*, Vol. 20, Issue 3, pp. 58-66, May 2003.

Authorized licensed use limited to: University of Texas at Austin. Downloaded on June 14,2010 at 00:31:47 UTC from IEEE Xplore. Restrictions apply.