A Novel Approach for Multi-Bit Error Correction in Memories

Jushwanth Xavier .X
Assistant Professor
Loyola Institute of Technology and Science

Benujah .B.R
Assistant Professor
Ponjesly College of Engineering

ABSTRACT

Due to advance technologies transistor size shrinks which makes the devices more vulnerable to noise and radiation effect. This affects the reliability of memories. Built-in current sensors (BICS) have been success in the case of single event upset (SEC). The process is taken one step further by proposing specific error correction codes to protect memories against multiple-bit upsets and to improve yield have been proposed. The method is evaluated using fault injection experiments. The results are compared with Hamming codes. The proposed codes provide a better performance compared to that of the hamming codes in terms of Single Event Upset. In the case of the Multi Bit Upset it provides better coverage in error deduction and correction.

General Terms

Error correction, VLSI

Keywords

Multi-bit error correction, Single event upset, hamming codes.

1. Introduction

CMOS scaling process provides high-density, low cost, low power, high-speed integrated circuits with a small noise margin. In very deep sub-micron technologies due to atmospheric neutrons and alpha particles the device's field-level reliability is severely impacted by single-event upset (SEU) and multi-bit event upset (MBU). Due to this features susceptible temporary faults will be increased [1]. Due to this not only memories but logic are also affected. When these particles hit the silicon bulk, they produce minority carrier, which produces voltage change at the nodes.

In combinational circuits soft error rate has drawn a major attention as the number of fault in the devices have increased significantly. Circuit latch up at output due to neutron effect have become second point, not many techniques cope with this problem. Effective solutions in protecting memories are also provided in [2]. Transient faults in space applications are potential consequences for the space craft that includes loss of information, functional failure of the craft [3]. Although SEUs are major problem, multiple-bit upset (MBU) has become important problem in the design memory devices. The probability of multiple errors due to technology shrinkage is given in [4] and [5]. As the size of the memories increases the probability of having multiple bits upset increases since large number of memory cells are used [6] and [7].

Packing and shielding cannot be effective against MBUs and SEUs since the neutrons can penetrate through the shield packages [5], [8].

Interleaving is the Common approach used in memory, in which the cell that belonging to the same logical word are placed at different positions during the design. The MBU errors are caused to the cells that are closer discussed in [9]. However this method cannot be used in larger memories because of the high accesses time, power consumption and floor plan discussed in [10]. Built-in current sensors (BICS) can deduct errors by detecting changes in the current as in [11], [12]. The protection can be optimized with the error correction codes (ECC) to cope up with MBUs. This is the objective of this paper proposing a new ECC to overcome MBUs.

2. Error correcting codes

Error correcting codes are widely used in protecting memories against the soft errors that are occurring due to the changes in the environment and the operating point of the devices. Hamming codes are widely used to protect memories against SEU because of the reduced area and performance. Hamming codes are used for single error deduction (SEC) and multiple error deduction. Hamming codes are capable of deducting up to two errors in a given code word. In order to improve the efficiency of the error correction, Triple modular redundancy (TMR) is used. But TMR uses poling methods that increase the area along with hamming code can correct only one error. Hence BICS are used along with ECC with a trade-off with area. Different methods are proposed that depends on redundancy that gradually increases the area.

Error deduction and correction in memories should be simple since accesses time is a major criteria. Due to high bandwidth used in memories in SOC applications the efficiency of repairing the memories decreases and redundant methods cannot be used. Examples of such applications are presented in [13] [14]. In order to cope up with the errors during the manufacturing processes certain times half of the device is used this is done by setting the MSB of the memory to be 0 or 1. Divide by half technique to cope with this problem has been proposed [15], to improve the efficiency of the memories novel techniques are required in the error correction. The technique that is used here gradually can correct more number of the errors with improving the overall system reliability.

3. Proposed technique

In this detection/correction scheme the message bits are arranged in array format. This is a combination of the parity codes and the hamming codes. The *n*-bit code word is divided into n_1 sub-words of width n_2 (i.e. $k=n_1*n_2$). A (n_1, $n2$) matrix is formed where n_1 and n_2 represent the numbers of rows and columns, respectively. For each of the n_1 rows, the check bits are added for single error correction/double error detection. Another n_2 bits are added as vertical parity bits.

M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	B_0	B_1	B_2
M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	B_3	B_4	B_5
M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	B_6	B_7	B_8
M_{24}	M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	B_9	B_{10}	B_{11}
P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7			
P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}			

Fig 1: 32-bit logical organization

The technique is explained by considering a 32-bit word length memory, which is divided into a matrix format as shown in Fig. 1, where n_1=4 and n_2=8, M_0 through M_{31} are the data bits, C_0 through C_{19} are the horizontal check bits, p_0 – p_7 are the vertical parity bits. Hamming codes are applied to each row. For an 8-bit data, 5 Hamming check bits are required. Hence 5 check bits are added at the end of the 8 bits.

As mentioned above the horizontal bits $P_0 – P_{15}$ are calculated using the ordinary parity generators. While the entire right side bits $B_0 – B_{11}$ are calculated as follows:

$$B_0 = M_1 \oplus M_3 \oplus M_5 \oplus M_7 \quad (1)$$

$$B_1 = M_1 \oplus M_2 \oplus M_3 \oplus M_7 \quad (2)$$

$$B_2 = M_0 \oplus M_1 \oplus M_3 \oplus M_4 \oplus M_6 \quad (3)$$

Similarly the parity codes are also generated

Accordingly, we calculate all check bits for all rows using $B_{New} = B_{j+(cb*o)}$ and $M_{new} = M_{l+(n2*o)}$, where *cb* is the position of check bit in the row, *o* is the row number where is the corresponding check bit's position in the first row and *i* is the corresponding data bit's position in this first row. A Hamming decoder is used to decode each row. Decoding is done in two steps. First, the horizontal check bits are calculated using the saved data bits and compared with the saved horizontal check bits. This procedure is called syndrome bit generation and B_1 is called syndrome bit of check bit B_1. Second, using syndrome bits S_i, the single error detection (SED)/double error detection (DED)/no error (NE) signals are generated for each row. If DED is activated (double error is detected in a row), we use the vertical syndrome bits SP_i and the saved value of the bit we can correct any single or double erroneous bits in each row using (7)

$$M_{icorrect} = (M_{ierr} \oplus 0) \oplus (DED_j * SB_n) \quad (7)$$

where M_{ierr} is the erroneous bit, O_i the decoder output corresponding to the erroneous bit i, DED_i is the DED signal of row j and SB_n the syndrome parity of the corresponding parity of the bit, e.g., for M_{10} , we have SP_2.

It is important to mention that if more than two errors are present in the code word, this technique can correct errors in any row assuming that no error in the same column . If only two errors occur, they these can be corrected without any restriction. Algorithm 1 shows the procedure of detection and correction in the proposed method which is applied on a code word M, where B'_i and P'_i are the check bits and the parity bits that are calculated using the saved data bits in the memory. These are then compared with saved memory check bits and parity bits to calculate the syndrome bits SB and *SP*.

Algorithm 1: code verification algorithm (M: data)

1: Read the saved data bits of M

2: Generate check bits using saved data bits ($B'_0 – B'_{11}$)

3: Generate syndrome bits of check bits ($SB_0 – SB_{11}$)

4: Generate parity bits using saved data bits ($P'_0 – P'_7$)

5: Generate syndrome bits of parity bits ($SP_1 – SP_7$)

6: Correct every saved bit if it is erroneous using (7)

7: Output the corrected word

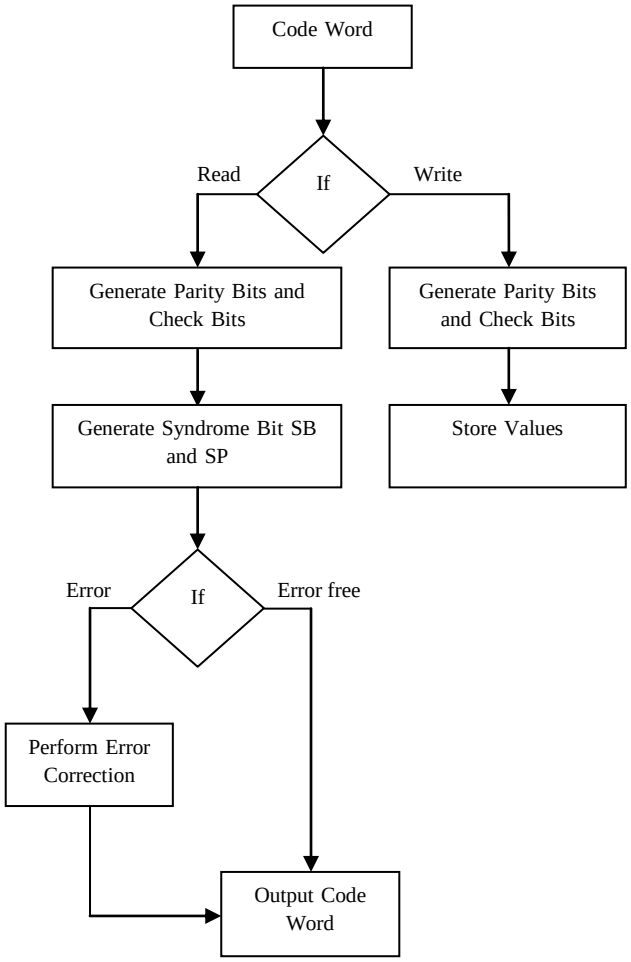

Fig 2: Flow Diagram

The read and write procedure for the memory with error correcting technique can be explained as follows. First each word in the modules is segmented into multiple bit segments. Then each *n* bit segment is encoded to *k* bit segment of (*k* - *n*) check bits. Algorithms 2 and 3 show the procedure for reading/writing words from/to a memory, respectively.

Algorithm 2: MEMORY READ

1. Read the word which contains the desired bits.
2. Correct for any errors.
3. Route the desired bits on the tree to the root node

Algorithm 3: MEMORY WRITE

1. Read the word which includes the desired bit.
2. Check for errors and correct them (if any)
3. Compare the value of the bit to be written against the value stored in the memory.

4. if bits are different then
5. Re-compute the check bits based on this new value.
6. Write back the data and the newly computed check bits
7. Else
8. Write back the data and the newly computed check bits
9. end if

Hence based on the algorithm the error deduction/correction is carried out

4. Simulation Results

The entire coding is done in verilog HDL and simulated. Fault injection is one of the key methods to estimate the error detection/correction capabilities of the circuits which utilize error detection and correction codes. Using a fault injection method, the coverage of the proposed technique was estimated. A thousand of faults were thrown and results were analysed. Compared to the previous methods the proposed method was able to deduct and correct up to eight errors in a row with a condition that no errors occur in the same column. Since this technique can correct only one error per row. Figure 3 shows the simulated fault injection method with three faults in data bits 0, 1 and 2 positions. And the result shows a successful error correction by using this method. Figure 4 shows the deduction and the correction coverage per code word of 8bits and found that the proposed technique

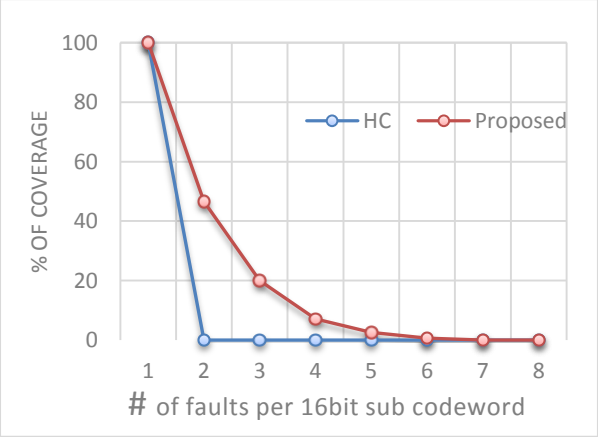

Fig 3: Fault coverage

Proves to be a more efficient method for multi-bit correction methods

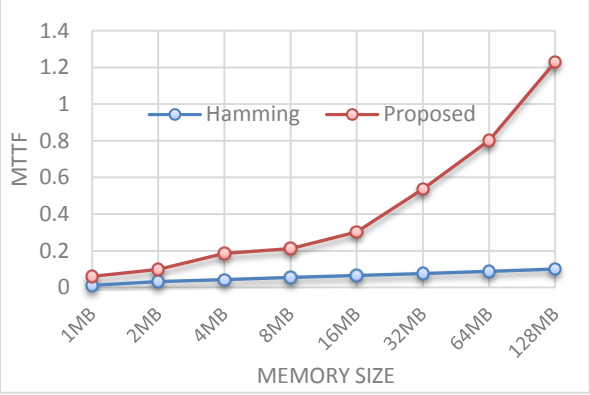

Fig 4: MTTF of 32bit code word

A memory chip protected with the proposed technique and Hamming Codes have been also described using Verilog language. Random faults were thrown into the memory and the METF for each technique was calculated. The METF of each technique was calculated using 15000 trials for each memory size, for more details, refer to [16]. We have used codeword sizes 32 bits. The results are portrayed in Fig. 4. Using the results obtained from METF using the expression expressed in equation (8) the mean time to fail analysis can be proceeded. For the 32bit code word a fault rate of about $\lambda = 10^{-5}$ where λ denotes the number of upsets that occur in a day table I gives the values that is obtained in the system during the analysis. The proposed technique provides a more efficient method and can provide better coverage.

TABLE I
MTTF IN DAYS OF PROPOSED TECHNIQUE WITH λ^{-5}

Type of protection	Memory Size		
	2Mb	16Mb	128Mb
Proposed codes	323.29	168.63	88.35
Hamming codes	15.35	5.47	1.95

$$MTTF = \frac{METF}{Memory\ Size * Fault\ Rate} \quad (8)$$

The redundant bits required for these techniques are tabulated in table II. The extra redundant bits required for providing the protection is 400% high compared to that of the hamming code where hamming code redundant is considered as 100%. Similarly for the 64bit word size the number of redundant bits that is required is about 500% where the redundant bits required for the hamming code is 100%.

TABLE II
REQUIRED REDUNDANT BITS

Type of protection	Word Size	
	32 bit	64 bit
Proposed method	28	40
Hamming Codes	7	8

Traditional memory repair techniques for yield improvement rely only on the addition of redundant rows and columns, which are then used to replace defective ones in the fabricated chip when necessary. As the number of redundant rows and columns is increased to allow higher repair capability, the fabrication yield also increases. However, since full size rows or columns must be used to replace defective ones, which usually contain only a few cells, this technique also increases the fabrication cost. Moreover, when all redundant rows and columns are exhausted, only a few chips can be used with degraded capacity by setting the most significant address bit to a constant value, due to the scattered distribution of defects in the array.
The technique proposed here aims to reduce the cost per chip and increase yield by using coding techniques that allows you to save some of the faulty memory chips with small defects instead of traditional redundant rows and columns. In the analysis of yield and cost per chip presented here, the following

assumptions have been adopted for the following simulations.

1) All defective memory chips have only spot defects, and no global defects, which are those defects affecting complete sections of a chip or wafer. For traditional techniques, when no redundant rows/columns are included in the chip, any single spot defect will result in the chip being discarded.

2) A 1024*32 bits memory array is used for area calculation, which has been obtained by modeling the chips

3) Each wafer can hold 1000 chips without any redundancy.

If redundant rows/columns are included on the chip, the number of chips per wafer is reduced. For example, if we add two redundant rows and two columns in each memory array the number of chips per wafer will be 969. (This is used only for the cost per chip.)

4.1. Yield Analysis

In order to confirm the yield benefits provided by our technique, we have performed several simulations of a production run for 1000 chips, with different numbers of defects per array and different quantities of redundant rows and columns in each run compared to our technique. In the yield analysis, the considered number of defects per chip in each simulated production run has been randomly distributed in the array, and simulations have been performed assuming the following scenario:

Chips are repaired using the coding techniques (Hamming and proposed) and chips with remaining defects after all redundant elements have been allocated are discarded. For each different simulation run, the yield has been calculated by dividing the effective number of chips that were considered good for sale after repair in each case by the total number of chips that are produced 1000 in our simulations. Using this criteria we evaluated the effectiveness of the proposed approach and calculated the yield for each technique.

4.2. Cost per Chip Analysis

In order to confirm the cost benefits provided by our technique, we have performed several simulations of a production run for 1000 chips, with different numbers of defects per array and using the same coding techniques. In the cost analysis, the considered number of defects per chip in each simulated production run has been randomly distributed in the array, and simulations have been performed assuming two different scenarios

Chips are repaired using the coding techniques (Hamming and proposed) and chips with remaining defects after all redundant elements have been allocated are discarded. For each different simulation run, the relative cost per chip has been calculated by dividing total number of chips that could be produced in the ideal scenario where no redundancy is used and no chips have defects (1000 in our simulations) by the effective number of chips that were considered good for sale after repair in each case

5. Conclusion

Here a high level error detection and correction method is introduced. The proposed protection code combines Hamming code and Parity code, so that multiple errors can be detected and corrected. The fault-injection based experimental results show that the proposed method provides better Detection and correction coverage than the Hamming codes. A 32 bit encoder and decoder are designed and simulated. By using the fault simulation method faults are forced for multiple bits by using forcing value in modelsim and results are verified. The code is able to deduct SEU/MBU and correct the errors. The research will be further extended to reduce the area and to improve the error correction of the proposed method.

6. References

[1] G. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Design of a fault tolerant solid state mass memory," *IEEE Trans. Reliab.*, vol. 52, no. 4, pp. 476–491, Dec. 2003.

[2] B. Cooke, "Reed Muller Error Correcting Codes," *MIT Undergraduate J. Math.*, vol. 1, pp. 21–26, 1999.

[3] P. A. Ferreyra, C. A. Marques, R. T. Ferreyra, and J. P. Gaspar, "Failure map functions and accelerated mean time to failure tests: New approaches for improving the reliability estimation in systems exposed to single event upsets," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 1, pp. 494–500, Jan. 2005.

[4] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 6, pp. 2586–2594, Dec. 2000.

[5] J. Karlsson, P. Liden, P. Dahlgren, R. Johansson, and U. Gunneflo, "Using heavy-ion radiation to validate fault-handling mechanisms," *IEEE Trans. Microelectron.*, vol. 14, pp. 8–23, 1994.

[6] R. Reed, M. Carts, P. Marshall, C. J. Marshall, O. Musseau, P. Mc- Nulty, D. Roth, S. Buchner, J. Melinger, and T. Corbiere, "Heavy ion and proton-induced single event multiple upset," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 6, pp. 2224–2229, Dec. 1997.

[7] N. Seifert, D. Moyer, N. Leland, and R. Hokinson, "Historical trend in alpha-particle induced soft error rates of the Alpha microprocessor," in *Proc. 39th Annu. IEEE Int. Reliab. Phys. Symp.*, 2001, pp. 259–265.

[8] S. Satoh, Y. Tosaka, and S. A. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's" *IEEE Electron Device Lett.*, vol. 21, no. 6, pp. 310–312, 2000.

[9] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code." in *Proc. IEEE VLSI Test Symp. (VTS)*, 2007, pp. 349–354.

[10] M. Nicolaidis, F. Vargas, and B. Courtois, "Design of built-in current sensors for concurrent checking in radiation environments," *IEEE Trans. Nucl. Sci.*, vol. 40, no. 6, pp. 1584–1590, Dec. 1993.

[11] J. Lo, "Analysis of a BICS-only concurrent error detection method," *IEEE Trans. Computers*, vol. 51, no. 3, pp. 241–253, 2002.

[12] S. K. Lu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," *IEEE Trans. Very Large Scale Integr. (VLSI) Systems*, vol. 14, no. 1, pp. 34–42, Jan. 2006.

[13] C. Argyrides, A. A. Al-Yamani, C. Lisboa, and L. C. D. K. Pradhan, "Increasing memory yield in future technologies through innovative design," in *Proc. 8th Int. Symp. Quality Electron. Des. (ISQED)*, Mar. 2009, pp. 622–626.

[14] C. Argyrides, H. Zarandi, and D. K. Pradhan, "Matrix codes: Multiple bit upsets tolerant method for SRAM memories", *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07*, Pp. 340–348.

[15] J. A. Maestro and P. Reviriego, "Study of the effects of MBUs on the reliability of a 150 nm SRAM device," in *Proc. 45th Annu. Des. Autom. Conf. (DAC)*, 2008, pp. 930–935.