IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 2, April-May, 2015
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Analysing The Power Overhead Of Decimal Matrix Code With Different Adder Architecture

[1]Gayathree.K, [2]David.S

PG Student, Department of Electronics and Communication Engineering,
Sri Eshwar College of Engineering, Coimbatore, India

Assistant professor, Department of Electronics and Communication Engineering,
Sri Eshwar College of Engineering, Coimbatore, India

***Abstract*—The data stored in the memory may be corrupted, when the memory is exposed to radiations. To prevent the data several error correction codes are used but they require complex encoder and decoder architecture. The decimal matrix code (DMC) minimizes these complexities compared to the existing codes such as PDS, built in current sensor etc and it also improves the reliability of the memory by increasing the error detection capability. In the proposed work two different adders namely ripple carry adder and hybrid adder are used in the DMC architecture and the power incase of each is analysed. The DMC architecture with hybrid adder consumes less power compared to the one with ripple carry adder.**

***Keywords*—Multiple Cell Upsets (MCUs), Error Correction Codes (ECCs), Decimal Matrix Code (DMC), Ripple Carry Adder (RC), Hybrid Adder (HA).**

I. INTRODUCTION

The probability of fault occurrence in the memories is increasing due to the scaling down of CMOS technology from micrometer to nanometer. This led to low power, low cost, high density and high speed integrated circuits. The failure rate of memory i.e, SRAM is thus gradually raised. Some of the error correction codes like matrix code, built-in current sensor with hamming and parity code, PDS etc are used to overcome the problem. The memory can be affected in two ways either an SEU or MCUs.

Reliability of the memory is a major issue. The data stored in the memory gets corrupted when the memory is exposed to external radiation like uv-rays, gamma rays etc. Then corrupted data can be corrected by reading the data and re-writing it on the memory location periodically. This operation is carried out by the encoder and decoder module of the error correction systems. However the encoder and decoder use complex architecture, which increases the area, delay and power overheads.

Hamming code is the most commoly used error correction codes but it possesses limited error detection capability. To overcome this extended hamming code is used but the error correction capability is limited eventhough the minimum distance is increased to 4. The single error detection and double adjacent error detection presented in [2], [3] achieves enhanced detection by performing selective shortening and reordering of the matrix. The difference set codes used in [4] is capable of detecting large number of errors but it consumes more time to decode. This affects the performance overheads of the memory. The triple error correction codes is compared with the single error correction codes in [5] and the results shows that TEC has better error correction capability and wider choice of word length.

Built-in current sensors [6] are a single error correction and double error correction code. The BICS combined with hamming code provides better performance compared to the BICS with the parity code. Matrix code [7], [8] divides the word into rows and columns where the hamming codes protect the bits per row and parity code protects the columns. The soft error in SRAM is analysed using Monte- Carlo Simulator in 22nm technology [9]. The ternary content addressable memory (TCAM) is more susceptible to soft error compared to the SRAM. In [10] a model is proposed to determine the scrubbling interval by predictive probabilistic failure rate analysis. The model in [11] showed the difference in the failure propabilities of various interleaving schemes. The interleaving technique cannot be applicable to content addressable memory. The SEU in CMOS SRAM with 130nm and 180nm technology is analysed using mono-energetic and quasi mono-energetic accelerators in [12].

The decimal matrix code [1] uses decimal addition/subtraction and ex-or operation to detect and correct errors present in the memory. Moreover DMC possess simple encoding and decoding architecture. The complexity is reduced by reusing the encoder and results in low power consumption. The DMC architecture with different adder like ripple carry adder and hybrid adder are proposed and the power consumption in both the cases are analysed. Ripple The DMC with hybrid adder seems to consume less power compared with the DMC with ripple carry adder.

II. DECIMAL MATRIX CODE

2.1 DMC architecture

The basic DMC architecture is described in Fig. 1. The information bit is fed to the encoder, where the information is protected by combining it with horizontal and vertical syndrome bits. This secured data is stored in memory.The memory may be exposed to MCUs which results in the corruption of the data. To obtain the original information from the corrupted data, the data is retrived and decoded. In the decoder the syndrome bits are obtained from the corrupted data and it will be compared with that of the syndrome bit calculated in the decoded.In this way error is detected and corrected.

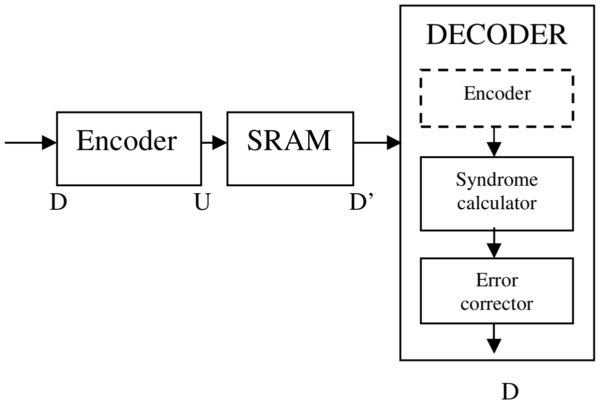


Fig. 1 DMC Architecture

The input stream is denoted as D, secured bit as U with horizontal and vertical syndrome bit, D' is the corrupted data from the memory.

2.2 Proposed Encoder

The flow of encoding process is shown in Fig. 4, in which the input , i.e N-bit word is first divided into K-symbols of m-bit each,where N=K×m. The symbol K is represented in the matrix form given by k1×k2 where k1 and k2 are rows and columns of the matrix. The horizontal syndrome bit is calculated using the decimal addition operation of selected symbols per row. The vertical syndrome bit is computed using exor operation of the bits.

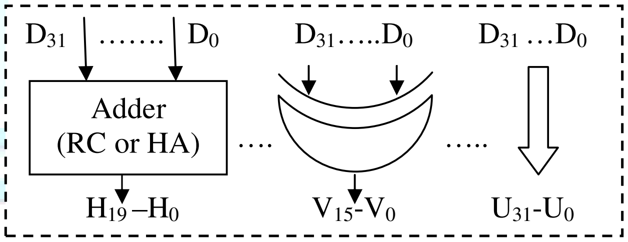


Fig. 2 32- bit DMC Encoder

Consider a 32-bit word which is divide into 8 symbols of 4-bit each is arranged in a 2×4 matrix format (k1=2 and k2 =4). A 32-bit DMC encoder module is shown in Fig. 2. The horizontal syndrome bit is computed by the following relation

$$H_4H_3H_2H_1H_0=D_3D_2D_1D_0+D_{11}D_{10}D_9D_8 \quad (1)$$

$$H_9H_8H_7H_6H_5=D_7D_6D_5D_4+D_{15}D_{14}D_{13}D_{12} \quad (2)$$

Similarly all the horizontal syndrome bits are calculated. The vertical syndrome bit is obtained using the following relation

$$V_0=D_0 \text{ xor } D_{16} \quad (3)$$

$$V_1=D_1 \text{ xor } D_{17} \quad (4)$$

Similarly V_2,V_3, etc are calculated using the equation mentioned.H_{19}-H_0 are the 20-bit horizontal syndrome bits,V_{15}-V_0 are the 16-bit vertical syndrome bit ,U_{31}-U_0 are the encoded bit and D_{31}-D_0 are the information bit.

Symbol 7 Symbol 2 Symbol 5 Symbol 0

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_{9}	D_{8}	D_{7}	D_{6}	D_{5}	D_{4}	D_{3}	D_{2}	D_{1}	D_{0}
D_{31}	D_{30}	D_{29}	D_{28}	D_{27}	D_{26}	D_{25}	D_{24}	D_{23}	D_{22}	D_{21}	D_{20}	D_{19}	D_{18}	D_{17}	D_{16}

Fig. 3 32-bit DMC Word Organization

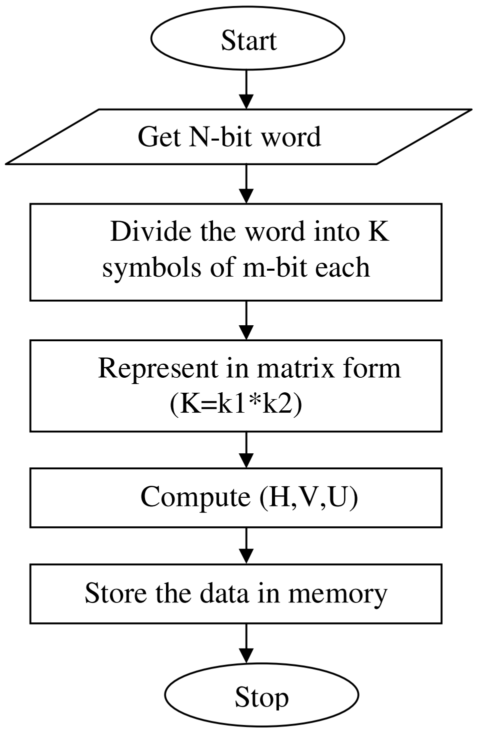

Start
Get N-bit word
Divide the word into K symbols of m-bit each
Represent in matrix form (K=k1*k2)
Compute (H,V,U)
Store the data in memory
Stop

Fig. 4 Flow of Encoding process

2.3 Proposed Decoder

A 32-bit DMC decoder module is shown in Fig. 5. During the read operation, the data from the memory is retrived. The D' is the data bit stored in the memory which may be corrupted. From the D' the redundant bits are generated using the encoder. The syndrome bits ΔH and S are calculated in the syndrome calculator block. The difference between the received redundant bit and the obtained syndrome bit is computed using the decimal integer subtraction.The occurrence of error is obtained using the given relation

$$\Delta H \sim S = 0 \qquad (5)$$

If the difference is equal to zero then there is no error in the received data but if the difference is non-zero then the data is corrupted and the error is detected.The locator locates error using vertical syndrome bits. The error corrector will correct the error by inverting the bits. The relation to correct the error is

$$D_{0CORRECT} = D_0 \text{ XOR } S_0 \qquad (6)$$

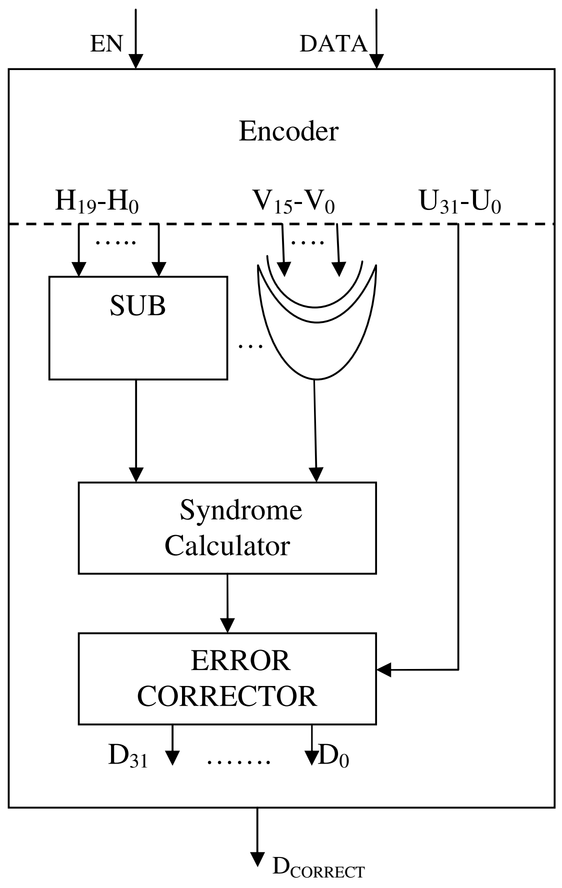


Fig. 5 32-bit DMC Decoder

Similarly for every bit D_i the error is corrected using (6), where i=1,2,3,4….31.

The number of error detected depends on the the (K,m) combination, K is number of symbols and m is the number of bits in each symbol. If k=2×2 and m=8 only 1-bit will be detected. If k=4×4 and m=2, 3-bit of error can be detected in a single row. If k=2×4 and m=4 upto 5-bit of error can be detected.The flow of decoding process is shown in Fig.6, in which the data corrupted is retrieved from the memory and the horizontal and vertical syndrome bits are obtained. Now calculate the new horizontal and vertical syndrome bits ΔH and S. Compare the two syndrome bit, if the difference is zero then there is no error. Thus the error is detected and corrected using the relation (6). To compute the difference in the syndrome calculator the decimal integer subtraction is carried out.

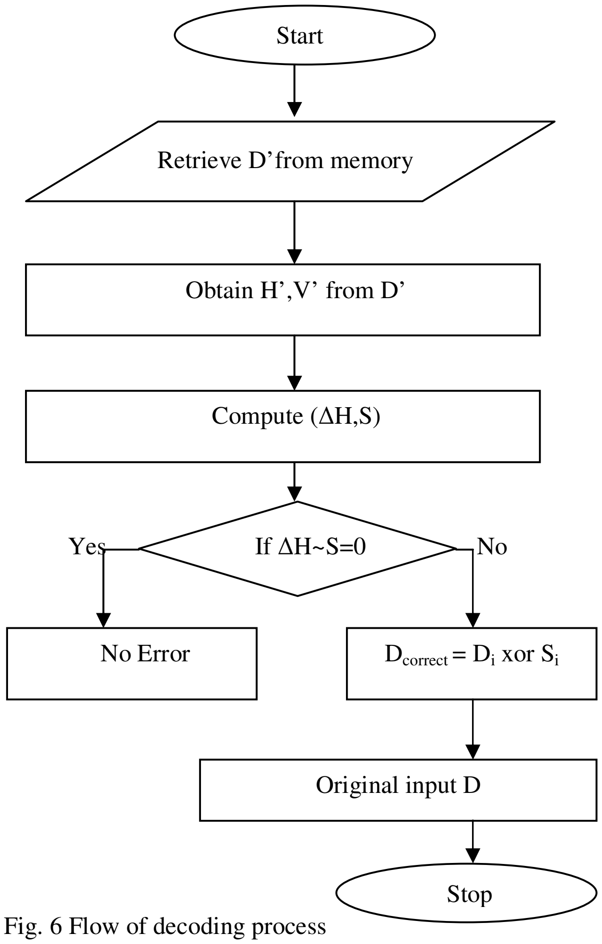


Fig. 6 Flow of decoding process

The encoder is reused in the decoder ,i.e a single encoder is used to perform both the encoding operation in case of encoder and syndrome bit calculation in case of decoding by which the area overheads is minimized. The selection of encoder is done using an enable signal ,the way how the encoder operation is selected is shown in table I.

TABLE I

Encoder Selection

Enable signal	Operational modes
0	Encoder
1	Syndrome calculator

III. ADDER ARCHITECTURE

3.1 Ripple Carry Adder

Ripple carry adder is a kind of high speed adders, used to improve the overall performance. RC adder is the combination of several 1-bit full adders that receive inputs and produce sum and carry at the output. In case of DMC architecture, the adder plays a prominent role in the calculation of syndrome bit which is inturn responsible for the fault tolerance capability of the memory. In the first model the multibit ripple carry adder is used in the DMC architecture.

Consider a 32-bit ripple carry adder shown in fig. 7, which is used to compute the horizontal syndrome bit H using the relation (1) and (2). In the 32-bit DMC architecture the 32-bit word is divided into 8 symbols of each 4-bit. During the horizontal syndrome bit computation, first any two symbols are selected and then they are fed as input to the RC adder,in which addition of two 4-bit symbols result in the 5-bit horizontal syndrome bit H. Similarly all the symbols are added and the syndrome bits are computed. Finally for a 32-bit word 20 horizontal syndrome bits are calculated.

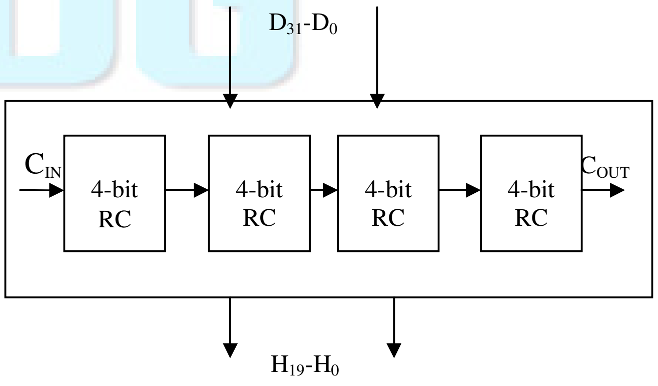


Fig. 7 32-bit Ripple carry adder

3.2 Hybrid Adder

Hybrid adder is a combination of several adders like carry look ahead, ripple carry adder, carry select and carry save adders etc. The hybrid adder is used mainly to gain the advantage of many adder inorder to have a better performance rather then using a single adder. In the second model hybrid adder which is the combination of carry look ahead adder and ripple carry adder is used as a multi-bit adder in the DMC architecture.

The block diagram of a 32-bit hybrid adder with carry look ahead adder and ripple carry adder is shown in fig. 8. In this the 32-bit word is divided into two 16-bit. The LSB 16-bit is fed to ripple carry adder and the MSB 16-bit is fed to the carry look ahead adder. The final result is obtained by concatenating the outputs of two adders.

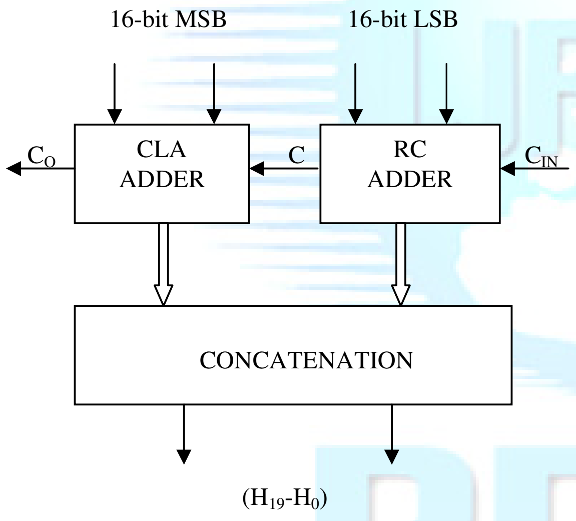


Fig. 8 Block diagram of Hybrid adder

CLA is the carry look ahead adder, while C_{IN} is the carry input and C_0 is the carry output and the input is the data D which is splitted into MSB and LSB respectively. In case of 32-bit DMC architecture with HA the 32-bit word is divided into symbols and these symbols are fed to CLA and RC adder to compute the horizontal syndrome bit H. Finally concatenating the result of two adders provides a 20 horizontal syndrome bit.

IV. POWER ANALYSIS OF DMC USING RC ADDER AND HA ADDER

4.1 Power Analysis

The power of DMC architecture using ripple carry adder and hybrid adder are compared and tabulated in table I. It also shows that the proposed decimal matrix code consumes less power than the existing codes like PDS,built-in current sensor with hamming code(BICS+HC),built-in current sensor with parity code(BICS+PR) etc. The Fig. 9 shows a bar chart with power in y-axis and various error correction codes in x-axis. From the fig. 9 it is found that the DMC with HA possess the least power of 52mW, while the reed muller code possess the maximum of 264.8Mw.

TABLE II

Power Analysis

ECCs	**Power (mW)**
DMC+HA adder	52
DMC+RC adder	54
PR+BICS	109.7
HC+BICS	139
PDS	221.2
RMC	264.8

The decimal matrix code with hybrid adder possesses lower power compared to the decimal matrix code with ripple carry adder. The hybrid adder uses the advantage of both RC and CLA adder and operates faster with lower delay overheads.

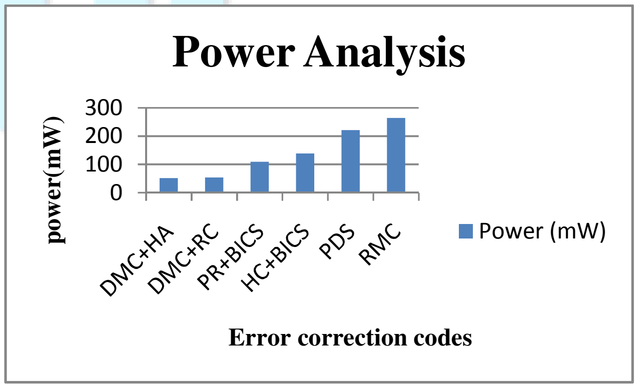
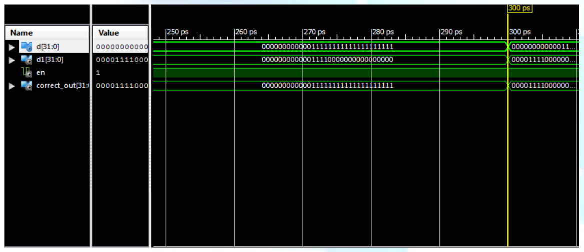


Fig. 9 Power Vs Various ECCs

V.SIMULATION RESULTS

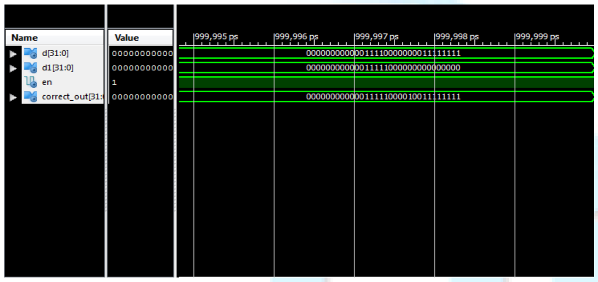
The proposed DMC is implemented in the VHDL,simulated with xilinx and modelsim. Testing is carried out using various inputs. This is an 180nm technology. The simulation result of DMC with ripple adder is shown in Fig.10. A 32-bit input d is given to the encoder and d1 is the error data and en is the enable signal and correct-out is the 32-bit corrected data with no error. The simulation result of DMC with hybrid adder is shown in Fig.11. Similar to the first architecture d is the 32-bit input ,d1 is the erroreous data and en is the enable signal. The error free output is represented as correct-out.

The power consumption of the two architecture is analysed in xilinx using XPower analyser. The values of power incase of DMC with RC is shown in Fig. 12.



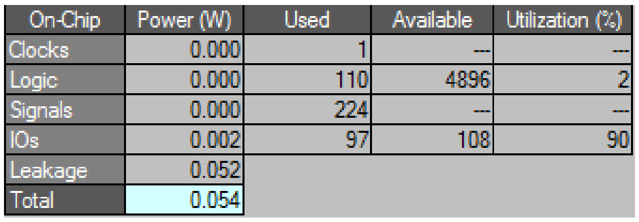
```
Name              Value
d[31:0]           00000000000   00000000000011111111111111111111   000000000000011...
d1[31:0]          00001111000   00000000000011110000000000000000   0000111100000000...
en                1
correct_out[31:   00001111000   00000000000011111111111111111111   0000111100000000...
```

Fig.10 Simulation result of DMC with RC adder



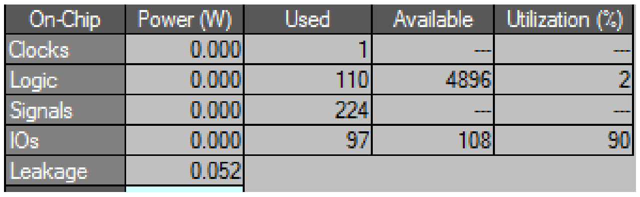
```
Name              Value
d[31:0]           00000000000   00000000000011111000000000111111
d1[31:0]          00000000000   00000000000011111000000000000000
en                1
correct_out[31:   00000000000   00000000000011111000010011111111
```

Fig.11 Simulation result of DMC with Hybrid adder



On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	110	4896	2
Signals	0.000	224	---	---
IOs	0.002	97	108	90
Leakage	0.052			
Total	0.054			

Fig.12 Power of DMC with RC adder



On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	110	4896	2
Signals	0.000	224	---	---
IOs	0.000	97	108	90
Leakage	0.052			

Fig. 13 Power of DMC with hybrid adder

Fig.13 provides the power value incase of DMC with hybrid adder. From the simulation result and power analysis it is found that the DMC with hybrid adder consumes 52mW which is 2mW less than that of the DMC with RC.

VI.CONCLUSION

The proposed decimal matrix code is a efficient error correction code in which the reliability and security of the memory is improved. The proposed code uses decimal algorithm which possess integer addition and subtraction which is simpler than the binary algorithm which performs bit wise logical operation. The binary algorithm uses only limited number of syndrome bit whereas the decimal matrix code uses more syndrome bit namely 20 horizontal syndrome bit and 16 vertical syndrome bit respectively which constitute a total of 36 bit. A single encoder is used in the DMC architecture that act as encoder as well as syndrome calculator,this will reduce the area overhead.

The decimal matrix code consumes less power than the other existing codes. The major component of DMC is multi-bit adder whose design will have a significant impact on the overall performance of the DMC system. The DMC with RC adder uses 54mW of power while the DMC with HA uses only 52mW of power. From the proposed models, it is found that the DMC with HA consumes power which is 2mW less than the one with RC adder. Further the power can be reduced by replacing the RC adder or HA adder with more advanced adder. The number of syndrome bit used to protect the data is more but the reduction in the number will inturn degrade the immunity of the memory.

REFERENCES

[1] Jing Guo and Liyi Xiao, "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code," *IEEE Trans on Very Large Scale Integration (VLSI) Systems, Vol. 22, No. 1*, January 2014.

[2] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement," *IEEE Trans. Device Mater. Rel., Vol. 14, No.1,* March 2014 .

[3] N. N. Mahatme, B. L. Bhuva, Y. P. Fang, and A. S. Oates, "Impact of strained-Si PMOS transistors on SRAM soft error rates," *IEEE Trans.Nucl. Sci.*, vol. 59, no. 4, pp. 845–850, Aug. 2012.

[4] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[5] P. Reviriego, M. Flanagan, and J. A. Maestro, "A (64,45) triple error correction code for memory applications," *IEEE Trans. Device Mater.Rel.*, vol. 12, no. 1, pp. 101–106, Mar. 2012.

[6] C. Argyrides, R. Chipana, F. Vargas, and D. K. Pradhan, "Reliability analysis of H-tree random access memories implemented with built in current sensors and parity codes for multiple bit upset correction," *IEEE Trans. Rel.*, vol. 60, no. 3, pp. 528–537, Sep. 2011.

[7] C. Argyrides, D. K. Pradhan, and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 3, pp. 420–428, Mar. 2011.

[8] C. A. Argyrides, P. Reviriego, D. K. Pradhan, and J. A. Maestro, "Matrix-based codes for adjacent error correction," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 4, pp. 2106–2111, Aug. 2010.

[9] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron induced soft error in SRAMs from an 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[10] S. Baeg, S. Wen, and R. Wong, "Minimizing soft errors in TCAM devices: A probabilistic approach to determining scrubbing intervals,"*IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 814–822, Apr. 2010.

[11] S. Baeg, S. Wen, and R. Wong, "Interleaving distance selection with a soft error failure model," *IEEE Trans. Nucl. Sci.*, vol. 56, no. 4,pp. 2111–2118, Aug. 2009.

[12] Y. Yahagi, H. Yamaguchi, E. Ibe, H. Kameyama, M. Sato, T. Akioka, and S. Yamamoto, "A novel feature of neutron-induced multi-cell upsets in 130 and 180 nm SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 1030–1036, Aug. 2007.