# ENHANCED DECIMAL MATRIX CODE FOR DETECTION AND CORRECTION OF CELL UPSETS IN SRAM

**Y.Madhusudhana[1], B. Suresh babu[2]**

*[1] II M.Tech, VLSI DESIGN, Dept of ECE, SRIT, Ananthapuramu, Andhra Pradesh, India*
*[2] Associate Professor, Dept of ECE, SRIT, Ananthapuramu, Andhra Pradesh, India*

---***---

**Abstract** - ***The technology advancements-scaling to smaller dimensions, higher integration densities, and lower operating voltages has come to a level that reliability of memories is put into jeopardy, not only in extreme radiation environments like spacecraft and avionics electronics, but also at normal terrestrial environments. SRAM memory failure rates are increasing significantly, therefore posing a major reliability concern for many applications.In order to make memory cells as fault-tolerant as possible, some error correction codes (ECCs) have been widely used to protect memories against soft errors for years. For example, the Reed–Solomon codes have been used to deal with multiple cell upsets (MCUs) in memories. But these codes require more area, power, and delay overheads since the encoding and decoding circuits are more complex in these complicated codes. More recently, a novel method using Decimal Matrix codes (DMCs) has been proposed to efficiently correct MCUs per word with a low decoding delay, in which one word is divided into multiple rows and multiple columns in logical. It uses decimal algorithm to protect information bits. The drawback of this method is that it requires more number of redundant bits and the correction capability is less.To overcome the above mentioned problem, some modifications are introduced in the structure of DMC. The proposed method can correct more bit errors with less number of redundant bits when compared to the existing DMC. Besides, the encoder-reuse technique (ERT) is proposed to minimize the area overhead of extra circuits (encoder and decoder) without disturbing the whole encoding and decoding processes. The experimental results showed that the proposed method has better protection level against large MCUs. In this project we used Modelsim for logical verification, and further synthesizing it on Xilinx-ISE tool.***

***Key words:*** *Radiation, mcu, dmc, reduntant bits, ert, hamming code.*

## 1. INTRODUCTION

Electronic space provided by silicon chips (semiconductor memory chips) or magnetic/optical media as temporary or permanent storage for data and/or instructions to control a computer or execute one or more programs. Two main types of computer memory are: one is Read only memory (ROM), smaller part of a computer's silicon (solid state) memory that is fixed in size and permanently stores manufacturer's instructions to run the computer when it is switched on; and the other is Random access memory (RAM), larger part of a computer's memory comprising of hard disk, CD, DVD, floppies etc., (together called secondary storage) and employed in running programs and in archiving of data. Memory chips provide access to stored data or instructions that is hundreds of times faster than that provided by secondary storage.Memories are affected by two types of errors. They are hard errors and soft errors. Hard errors are caused due to fabrication defects in the memory chip and cannot be corrected once they start appearing. Soft errors on the other hand are caused predominantly by electrical disturbances.

### SOFT-ERRORS

Soft-errors [5] are caused predominantly by electrical disturbances. A soft-error occurs when a radiation event causes enough of a charge disturbance to reverse or flip the data state of a flip-flop or memory cell. The error is soft because the circuit/device itself is not permanently damaged just device fall into an incorrect state or goes into an unexpected reset.A single event upset (SEU) is a change of state caused by ions or electro-magnetic radiation striking a sensitive node in a micro-electronic device, such as in a microprocessor, semiconductor memory, or power transistors. The state transformation remains a result of the free charge made through ionization in or close towards an vital node of a logic element. The fault in device output or process affected as a outcome of the raid is called an SEU or a soft error.The SEU itself is not considered permanently damaging to the transistor's or circuits' functionality unlike the case of single event latch up (SEL), single event gate rupture (SEGR), or single event burnout (SEB). These remain entirely samples of a common class of emission effects in electronic devices called distinct event effects. Terrestrial SEU arise due to cosmic elements striking by atoms in the atmosphere, making cascades or showers of neutrons as well as protons, which in order might relate by electronics. At deep submicron meter geometries, this affects semiconductor devices in the atmosphere. In space, high energy ionizing elements be as part of the natural background, mentioned to as galactic cosmic rays (GCR). Solar particle measures as well as high energy protons stuck in the Earth's magnetosphere worsen the difficult.

Inferior atmospheric neutrons made in cosmic rays can too remain of energies accomplished of creating SEUs in electronics happening aircraft flights finished the poles otherwise on great altitude. Bit quantities of radioactive components in chip packages too lead to SEUs.

**HARD ERRORS**

It is an error occurrence in a computer system that is caused by the failure of a memory chip. Hard errors can appear like chip-level soft errors, but the difference is that the hard error is not rectified when the computer is rebooted. Hard errors are mainly caused at the time of fabrication that cannot be recovered.

**ERROR CONTROL CODING**

The general idea for achieving error detection and correction (ECC) [7,8] is to add some redundancy towards message, which receivers can practice towards patterned steadiness of carried message, as well as to recuperate data resolute towards remain tarnished. Error-detection and correction schemes can be either systematic or non-systematic: In a systematic scheme, the transmitter sends the original data, and attaches a fixed number of check bits (or parity data), which are derived from the data bits by some deterministic algorithm. If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits; if the values do not match, an error has occurred at some point during the transmission. In a system that practices non-systematic code, the unique data be distorted into an encrypted data that consumes as a minimum as several bits as the unique data.Good fault control enactment needs the system towards remain designated founded on the features of the communication channel. Common channel replicas embrace memory-less replicas anywhere faults happen erratically as well as by a firm possibility, then active replicas wherever faults happen mainly in bursts. Consequently, error-detecting and correcting codes can be generally distinguished between random-error detecting or correcting and burst-error- detecting or correcting. Certain codes can also be fit on behalf of a combination of arbitrary faults as well as burst faults.Certain modest codes can perceive but not precise faults; others can notice as well as precise one or more faults. This paper reports one Hamming code that can correct a single-bit error as well as notice a double-bit error.

## 2.EXISTED METHODS

In this chapter we are introducing an existing Decimal Matrix Code (DMC) [1]. It is based on divide-symbol and arrange-matrix logic to provide enhanced memory reliability. The DMC utilizes decimal algorithm (decimal integer addition and decimal integer subtraction) to detect errors. The advantage of using decimal algorithm is that the error detection capability is maximized so that the reliability of memory is enhanced. Besides, the Encoder-Reuse Technique (ERT) is proposed to minimize the area overhead of extra circuits (encoder and decoder) without disturbing the whole encoding and decoding processes, because ERT uses DMC encoder itself to be part of decoder.

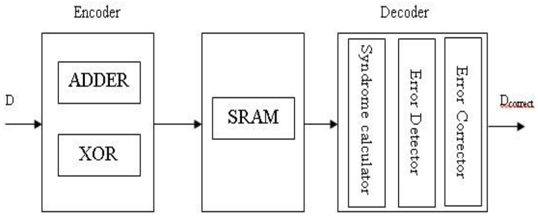

**Fig:** Block diagram of existing method

**DECIMAL MATRIX CODE ENCODER**

In the existing DMC, first, the divide-symbol and arrange-matrix ideas are performed, i.e., the $N$-bit word is divided into $k$ symbols of $m$ bits ($N = k \times m$), and these symbols are arranged in a $k_1 \times k_2$ 2-D matrix ($k = k_1 \times k_2$, where the values of $k_1$ and $k_2$ represent the numbers of rows and columns in the logical matrix respectively). Second, the horizontal redundant bits $H$ are produced by performing decimal integer addition of selected symbols per row. Here, each symbol is regarded as a decimal integer. Third, the vertical redundant bits $V$ are obtained by binary operation among the bits per column. It should be noted that both divide-symbol and arrange-matrix are implemented in logical instead of in physical. Therefore, the DMC does not require changing the physical structure of the memory.To explain the DMC scheme, we take a 32-bit word as an example, as shown in Fig 3.2. The cells from $D_0$ to $D_{31}$ are information bits. This 32-bit word has been divided into eight symbols of 4-bit. $k_1 = 2$ and $k_2 = 4$ have been chosen simultaneously. $H_0$-$H_{19}$ are horizontal check bits; $V_0$ through $V_{15}$ are vertical check bits. However, it should be mentioned that the maximum correction capability (i.e., the maximum size of MCUs can be corrected) and the number of redundant bits are different when the different values for $k$ and $m$ are chosen. Therefore, $k$ and $m$ should be carefully adjusted to maximize the correction capability and minimize the number of redundant bits. For example, in this case, when $k = 2\times2$ and $m = 8$, only 1-bit error can be corrected and the number of redundant bits is 40. When $k = 4 \times 4$ and $m = 2$, 3-bit errors can be corrected and the number of redundant bits is reduced to 32. However, when $k = 2 \times 4$ and $m = 4$, the maximum correction capability is up to 5 bits and the number of redundant bits is 36. In this thesis, in order to enhance the reliability of memory, the error correction capability is first considered, so $k = 2 \times 4$ and $m = 4$ are utilized to construct DMC. The horizontal redundant bits $H$ can be obtained by decimal integer addition as follows:

$H_4 H_3 H_2 H_1 H_0 = D_3 D_2 D_1 D_0 + D_{11} D_{10} D_9 D_8$ & $H_9 H_8 H_7 H_6 H_5 = D_7 D_6 D_5 D_4 + D_{15} D_{14} D_{13} D_{12}$ and similarly for the horizontal redundant bits

$H_{14}\ H_{13}H_{12}\ H_{11}\ H_{10}$ and $H_{19}\ H_{18}\ H_{17}H_{16}\ H_{15}$, where "+" represents decimal integer addition.

For the vertical redundant bits $V$, we have

$V_0 = D_0 \oplus D_{16}$

$V_1 = D_1 \oplus D_{17}$

and similarly for the rest vertical redundant bits.The encoding can be performed by decimal and binary addition operations.

### DECIMAL MATRIX CODE DECODER

To obtain a word being corrected, the decoding process is required. For example, first, the received redundant bits $H_4\ H_3\ H_2\ H_1\ H_0$ and $V_0 - V_3$ are generated by the received information bits $D$. Second, the horizontal syndrome bits $H_4\ H_3\ H_2\ H_1\ H_0$ and the vertical syndrome bits $S_3 - S_0$ can be calculated as follows:

$\blacktriangle H_4\ H_3\ H_2\ H_1\ H_0 = H_4\ H_3\ H_2\ H_1\ H_0 - H_4 H_3 H_2 H_1 H_0$

$S_0 = V_0 \oplus V_0$

and similarly for the rest vertical syndrome bits, where "–" represents decimal integer subtraction, "⊕" represents Exclusive OR operation and "▲" is the difference symbol.

When $H_4\ H_3\ H_2\ H_1\ H_0$ and $S_3 - S_0$ are equal to zero, the stored codeword has original information bits in symbol 0 where no errors occur. When $H_4\ H_3\ H_2\ H_1\ H_0$ and $S_3 - S_0$ are nonzero, the induced errors (the number of errors is 4 in this case) are detected and located in symbol 0, and then these errors can be corrected by

$$D_{0\text{correct}} = D_0 \oplus S_0$$

The DMC decoder is depicted in Fig 3.1, which is made up of the following sub-modules, and each executes a specific task in the decoding process: syndrome calculator, error locator, and error corrector. It can be observed from this figure that the redundant bits must be recomputed from the received information bits $D$ and compared to the original set of redundant bits in order to obtain the syndrome bits $H$ and $S$. Then error locator uses $H$ and $S$ to detect and locate which bits some errors occur in. Finally, in the error corrector, these errors can be corrected by inverting the values of error bits.When the decimal algorithm is used to detect errors, these errors can be detected so that the decoding error can be avoided. The reason is that the operation mechanism of decimal algorithm is different from that of binary. The detection procedure of decimal error detection using the proposed structure is shown in Fig 3.3. First of all, the horizontal redundant bits $H_4\ H_3\ H_2\ H_1\ H_0$ are obtained from the original information bits in symbols 0 and 2 according to equation.

$$\begin{aligned} H_4\ H_3\ H_2\ H_1\ H_0 &= D_3\ D_2\ D_1\ D_0 + D_{11}\ D_{10}\ D_9\ D_8 \\ &= 1100 + 0110 \\ &= 10010 \end{aligned}$$

When MCUs occur in symbol 0 and symbol 2, i.e., the bits in symbol 0 are upset to "1111" from "1100" ($D_3\ D_2\ D_1\ D_0$ = 1111) and the bits in symbol 2 are upset to "0111" from "0110" ($D_{11}\ D_{10}\ D_9\ D_8$ = 0111). During the decoding process, the received horizontal redundant bits $H_4\ H_3\ H_2\ H_1\ H_0$ are first computed, as follows:

$$\begin{aligned} H_4\ H_3\ H_2\ H_1\ H_0 &= D_{11}\ D_{10}\ D_9\ D_8 + D_3\ D_2\ D_1\ D_0 \\ &= 0111 + 1111 \\ &= 10110 \end{aligned}$$

Then, the horizontal syndrome bits $H_4\ H_3\ H_2\ H_1\ H_0$ can be obtained using decimal integer subtraction

$$\begin{aligned} \blacktriangle H_4 H_3 H_2 H_1 H_0 &= H_4 H_3 H_2 H_1 H_0 - H_4 H_3 H_2 H_1 H_0 \\ &= 10110 - 10010 \\ &= 00100 \end{aligned}$$

The decimal value of $H_4\ H_3\ H_2\ H_1\ H_0$ is not "0," which represents that errors are detected and located in symbol 0 or symbol 2. Subsequently, the precise location of the bits that were flipped can be located by using the vertical syndrome bits $S_3 - S_0$ and $S_{11} - S_8$. Finally, all these errors can be corrected. Therefore, based on decimal algorithm, the proposed technique has higher tolerance capability for protecting memory against MCUs.

## 3. PROPOSED METHOD

The implementing method is a modified version of the existing DMC method. Here, the encoder block is modified i.e., the adder block is replaced by Hamming encoder which enhanced the correction capability and also reduced the number of redundant bits.

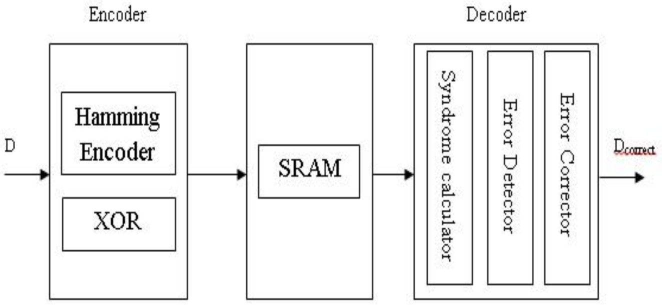

**Fig:** Block diagram of Proposed method

### MODIFIED-DECIMAL MATRIX CODE

The proposed block diagram of modified- DMC is shown in Fig 4.1. Here, the encoding process may be interpreted as writing the data into SRAM cells and the decoding process as reading the data from SRAM cells. The information bits D are fed to the DMC encoder which computes the horizontal check bits H and vertical check bits V. Once the encoding process is completed, the obtained modified-DMC code is stored in SRAM cells. Along with these bits, a copy of information bits D is stored in the register U. Now, when these cells are exposed to radiations, MCUs may occur. These MCUs are corrected in the decoding process. The proposed protection code utilizes decimal algorithm to detect errors. Since the decoder uses ERT technique, the area overhead of extra circuits is also reduced significantly.

### MODIFIED-DECIMAL MATRIX CODE ENCODER

The proposed modified-DMC encoder is similar to existing DMC encoder but employs Hamming codes for calculating horizontal redundant bits. The circuit of encoder using Hamming encoder and XOR gates is shown in Fig 4.2. First, an N-bit information word is taken as the source data. This N-bit word is divided into k-symbols of m-bits each which gives N = kxm. These k-symbols are arranged systematically in a $k_1xk_2$ 2-D matrix form such that $k = k_1xk_2$. Here $k_1$ indicates number of rows and $k_2$ indicates number of columns. Second, the horizontal check bits are computed by employing Hamming codes for each symbol of first row. Here, each symbol is considered as a decimal integer. Third, the vertical check bits are computed by employing binary exclusive OR operation for every column.

To understand the proposed Modified-DMC scheme, the logical implementation of a 32-bit DMC code is explained in Fig 4.3. The cells numbered from $D_0$ to $D_{31}$ are information bits. This 32-bit word (N = 32) is divided into eight symbols (k = 8) of 4 bits (m = 4) each. The cells numbered from $P_0$ to $P_{11}$ are horizontal check bits and from $V_0$ to $V_{15}$ are vertical check bits. The maximum correction capability and the number of redundant bits are different when different values for k and m are chosen. For example, if k = 2x4 and m = 4, it can correct maximum of 16-bit errors and the number of redundant bits are 28. Therefore, the values of k and m should be selected carefully.The horizontal redundant bits are calculated for all the symbols in only one row. The following equations represent the Hamming equations to calculate the horizontal check bits.

$P_0 = D_3 \wedge D_1 \wedge D_0$

$P_1 = D_3 \wedge D_2 \wedge D_0$

$P_2 = D_3 \wedge D_2 \wedge D_1$ …….and so on

The following equations are used to compute the vertical check bits.

$V_0 = D_0 \wedge D_{16}$

$V_1 = D_1 \wedge D_{17}$ ……… and so on

The symbol " **^** " indicates binary exclusive-OR operation.

### MODIFIED-DECIMAL MATRIX CODE DECODER

In the decoding process, the first step is to detect the errors if any and correct them accordingly. Here, the encoder re-use technique is employed. The received information bits D' are applied to the in-built encoder block in the decoder circuit to obtain the horizontal check bits $P_0'$ to $P_{11}'$ and vertical check bits $V_0'$ to $V_{15}'$.The schematic of the decoding circuit is shown in Fig 4.4.

The decoding process goes through a step by step process i.e., syndrome calculator, error-locator and error corrector. The decimal integer subtraction is used to compute horizontal syndrome bits and exclusive OR operation to compute vertical syndrome bits. The non-zero horizontal syndrome bits indicates error detection and the non-zero vertical syndrome bits gives the location of errors. These errors are corrected by error corrector block using XOR operations.

The horizontal syndrome bits are obtained as follows:

$P_0P_1P_2D_3D_2D_1D_0 = P_0P_1P_2D_3D_2D_1D_0' - P_0P_1P_2D_3D_2D_1D_0$

$P_3P_4P_5D_7D_6D_5D_4 = P_3P_4P_5D_7D_6D_5D_4' - P_3P_4P_5D_7D_6D_5D_4$

….and so on

The vertical syndrome bits are obtained as follows:

$S_0 = V_0' \wedge V_0$

$S_1 = V_1' \wedge V_1$ …….and so on

The errors can be corrected by $D_{0correct} = D_0' \wedge S_0$ … and so on In OMP dictionary atoms taken once never taken again; there by reduces the total algorithm complexity. Compared with general natural images, the fingerprint images have simpler structure. They are only composed of ridges and valleys. In the local regions, they look the same.

## 4.SIMULATION RESULTS

The encoder and decoder blocks are coded using VHDL language. The complete system is implemented for 32-bit data. The simulation results are obtained using Modelsim for both existing DMC and Modified-DMC methods.

### EXISTING DMC

Here a 32-bit data is encoded using existing DMC. Then, a 32-bit fault data is injected. Then, the corresponding errors are detected and corrected. The Fig 5.1 shows the simulation output of existing DMC method. The corresponding data used are given below. The positions where the errors are forced are also given. The corrected data is obtained.

Input data

D[31:0] : 1111 0101 1010 1111 1111 0110 1010 1100

Fault injected data

MCU[31:0] : 1111 0101 1010 1111 1111 0111 1010 0011

Errors are introduced at positions : 0, 1, 8

Corrected data

$D_{correct}$[31:0] : 1111 0101 1010 1111 1111 0110 1010 1100

### MODIFIED-DMC

Here a 32-bit data is encoded using Modified-DMC. Then, a 32-bit fault data is injected. The positions where the errors are forced are also given. Then, the corresponding errors are detected and corrected. The Fig 5.3 shows the simulation output of implementing Modified-DMC method. From the simulation waveform we can see that the Modified-DMC method effectively corrects the type 3 and type 4 errors.

Input data

D[31:0] : 1111 0101 1010 1111 1111 0110 1010 1100

Fault injected data

MCU[31:0] : 1111 1001 1010 1111 1101 0110 1010 0011

Errors are introduced at positions : 0, 1, 2, 3, 13, 26, 27

Corrected data

$D_{correct}$[31:0] : 1111 0101 1010 1111 1111 0110 1010 1100

## CONCLUSION

In this thesis, modified-DMC was proposed to protect memories from radiation induced errors. The proposed method uses a combination of Hamming encoder and decimal algorithm which allows detection and correction of large MCUs. Also, the use of Encoder re-use technique reduced the area overhead of extra circuits. The obtained results has reduced the number of redundant bits by 22.22%, reduced the area of the design by 11.87 %, reduced power consumption by 18.44% and reduced the delay by 21.38%. These results show that the proposed method has a superior protection level against multiple cell upsets in memory and provides more reliable data than the existing method.

## FUTURE SCOPE

In this design, the adder block is replaced with Hamming code block in the encoder structure which in turn reduced the number of horizontal redundant bits. Even though it enhanced the correction capability, it still uses a large amount of redundant bits. This thesis can further be researched in the decoder structure to reduce the vertical redundant bits keeping a higher correction capability.

## REFERENCES

1.Jing Guo, Liyi Xiao, “Enhanced memory reliability against multiple cell upsets using decimal matrix code,” IEEE Transactions 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., to be published.

3.R. Naseer and J. Draper, “Parallel double error correcting code design to mitigate multi-bit- upsets in SRAMs,” in Proc. 34th Eur. Solid-State Circuits, Sep. 2008,, pp. 222-225.

4.G. Neuberger, D. ?L. Kastensmidt, and R. Reis, “An automatic technique for optimizing Reed-Solomon codes to improve fault tolerance in memories,” IEEE Design Test Comput., vol. 22, no; 1, pp; 50-58, Jan.-Feb. 2005.

5.R. C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies,” IEEE Trans. Device Mater. Reliab., vol. 5, no. 3, pp. 301-316, 2005.

6.R. G. Gallager, “Low-density parity-check codes”, IRE Trans. Information Theory, vol. IT-8, no. 1, pp. 21–28, January 1962.

7.Wakerly, J. F. Error detecting codes, self-checking circuits and applications. New York: North-Holland, 1978.

8.A. D. Houghton, The Engineer’s Error Coding Handbook. London, U.K.: Chapman and Hall, 1997.

9.S. Baeg, S. Wen, and R. Wong, “Interleaving distance selection with a soft error failure model,” IEEE Trans. Nucl. Sci., vol. 814-822, Apr. 2010.

10.Shih-Fu Liu, Pedro Reviriego and Juan Antonio 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.

11.Hill, Raymond. A First Course in Coding Theory. Clarendon Press,1986.

12.D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes”, Electronics Letters, vol. 32, no. 18, pp.1645–1646, March 1997.

13.R. J. McEliece, The Theory of Information and Coding. Cambridge, U.K.: Cambridge University Press, 2002.

M. Sipser and D. Spielman, “Expander codes,” IEEE Trans. Inf. Theory, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

## BIOGRAPHIES

**Y.MADHUSUDHANA** received the B.Tech degree in Electronics & Communication engineering. Presently he is doing his M.tech in VLSI DESIGN from SRIT College, Ananthapuramu.

**B.SURESH BABU** working as Associate Professor in Dept of ECE, at SRIT College, Ananthapuramu received the B.Tech degree and M.Tech degree in Electronics & Communication engineering..