Original scientific paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 46, No. 4(2016), 257 – 266

Radiation Induced Multiple Bit Upset Prediction and Correction in Memories using Cost Efficient CMC

A.Ahilan[1], P. Deepa[2]

[1]Fulltime research scholar, GCT Coimbatore
[2]Assistant Professor, GCT Coimbatore

Abstract: This paper presents a cost efficient technique to correct Multiple Bit Upsets (MBUs) to protect memories against radiation. To protect memories from MBUs, many complex error correction codes (ECCs) were used previously, but the major issue is higher redundant memory overhead. The proposed method called counter matrix code (CMC) utilizes combinational ones counter and parity generator with less redundant memory overhead. CMC based on error predictor predicts the exact number of upsets before the actual error detection and correction process. The proposed technique uses Encode-Compare for minimizing the cost and increase the speed of the decoding process. The results are compared to the well-known codes such as CRC, Hamming and other matrix codes. The obtained results show that the correction coverage per cost (CCC) of the proposed scheme is higher than other traditional techniques. The mean time to repair (MTTR) of the proposed scheme is 3 times reduced than Xilinx cyclic redundancy check (CRC) + Reload technique for 100% correction coverage. At the same time MTTR of the proposed scheme is 0.3 ms, 0.2 ms and 1.8 ms less than I3D, DMC and MC, respectively with improved correction coverage.

Keywords: Multiple bit upsets (MBUs); memories; ones counter; parity codes; mean time to repair (MTTR)

Napoved in korekcija s sevanjem povzročenih večbitnih napak v pomnilnikih z uporabo učinkovitih CMC kod

Izvleček: Članek predstavlja učinkovito metodo korekcije večbitih napak (MBU) za zaščito pomilnikov pred sevanjem. V preteklosti so se za zaščito pomnilnikov uporabljale številne kompleksne metode popravljanja napak, ki pa so zahtevale veliko spominskega prostora. Predlagana metoda CMC združuje števec in generator paritete z manjšo zahtevo po redundančnem spominu. CMC napove natančno število napak pred dejansko detekcijo in korekcijo. Rezultati so primerjani z ostalimi metodami kot so: CRC, Hamming in druge. Rezultati izkazujejo učinkovitejšo korekcijo kot konvencionalne metode, pri čemer je povprečen čas korekcije 3 krat krajši kot pri Xilinx CRC tehniki. Istočasno je MTTR 0.3 ms, 0.2 ms in 1.8 ms krajši od I3D, DMC in MC.

Ključne besede: večbitne napake (MBUs); pomnilniki; pariteta ; števec; parity codes; povprečni čas korekcije (MTTR)

** Corresponding Author's e-mail: listentoahil@gmail.com*

1 Introduction

Today Electronic Design Automation (EDA) industries aims to make reliability the next level of radiation protection by drawing on advances in fault tolerant techniques to protect CMOS memory chips and promoting the protected memory chips to space and safety critical applications. SRAM memories are mainly utilized by reconfigurable devices like field reprogrammable gate arrays (FPGAs) and recent programmable system on chips (SoCs). Recently the usages of SRAM memories are increased and occupied more than 90% of chip area in modern SoCs [1-3]. These SRAM memories are disturbed by soft errors and distresses system reliability and sustainability [4-5]. Minimum transistor size and increased memory density due to technology scaling are becoming increasingly susceptible to multiple bit up-

sets (MBUs) [6]. The largest MBUs size observed in the neutron induced experiment is 24 bits [7]. For smaller nanometer technologies, this count of MBU size is even more [6]. This status evidently shows the significance of protecting SRAM memories against MBU incidents.

Several proven techniques have been addressed to protect SRAM memories from radiation induced soft errors in FPGA configuration frames. Xilinx design flow consisting single event upset (SEU) mitigation step to cope single bit soft errors [18]. In addition to that Xilinx offers a two adjacent erroneous bits correction using IP block as a soft error alleviation controller based on global cyclic redundancy check (CRC) and error correction coding (ECC) technique [19]. The most common and efficient approach to preserve a good level of reliability for memory words is to use ECCs. The widely used ECC for memory protection is Hamming and odd weight codes against radiation induced soft errors due to their ability to mitigate single bit upsets (SBUs) practically with reduced energy and area overhead [8], [9]. On the other hand, single charged particle can provoke MBUs in the memory words and these MBUs are not corrected by these single bit correctable ECCs. However, there are highly developed ECCs such as Reed–Solomon codes [15], Reed–Muller code [10] and punctured difference set (PDS) codes [16] have been used to mitigate MBUs in memories. But the encoding and decoding steps are more complex to cope with MBUs in these highly developed codes. More over this is achieved at the expense of high area, delay and power consumption.

In matrix code (MC) [11], two errors are corrected based on Hamming and vertical syndrome bits in all cases. Recently DMC proposed by Jing Guo et.al to correct MBU with high reliability, but it uses more redundant bits. For 32 bit memory word, 36 numbers of redundant bits are needed to correct MBU in DMC. This extra bits occupy more area in memory chip [12]. Parallel error correction code has been presented to correct MBU's with huge area overhead [13]. More recently, in [14], 2-D ECCs such as 2-D SHMC (Symbolic Hamming Matrix Code) and 2-D RMC (Reconfigurable Matrix Code) has been proposed to efficiently mitigate MBUs of 32-bit memory word. The advantage of these codes is that the delay is minimized due to the Encode-Compare mechanism instead of Decode-Compare mechanism. In [22], an approach that combines interleaved 3-D parity technique (I3D) with erasure code has been conceived to be applied at architectural level. It uses horizontal, vertical and diagonal parity bits to detect MBUs and erasure codes for MBU correction. The results achieved from this approach shown that additional recovery time needed to correct MBUs over other codes. Based on the combinational ones counter and parity code, preliminary version of algorithm has been proposed for MBU error prediction and error correction in SRAM [17].

In the proposed work, both intra and inter word error detection and correction and error prediction are introduced by combinational counting operation. The redundant bits used for the detection and correction are computed from the outputs of row and column counters. Computing redundant bits from group of words reduces the redundant memory overhead. This work uses Encode-Compare instead of Decode-Compare mechanism in decoder for reducing the delay overhead.

The presentation of this work can be divided into five sections. In section II, the proposed CMC is introduced and its encoder and decoder architectures are given with sample calculations. Section III discusses the correction coverage and overhead analysis of the various MBU mitigation methods. Conclusions and future work ideas are given in Section IV.

2 Proposed counter matrix code

In this section, CMC encoding and decoding algorithm is proposed to predict and correct the MBUs and the VLSI architectures for encoder and decoder are presented. The proposed CMC based encoding and decoding algorithm appears to lend itself to detect both

Table 1: 128-bit logical organization of CMC

S.No	Symbol8	Symbol7	Symbol6	Symbol5	Symbol4	Symbol3	Symbol2	Symbol1	H_{CC}	H_{PC}
0	$B_0^{(31-28)}$	$B_0^{(27-24)}$	$B_0^{(23-20)}$	$B_0^{(19-16)}$	$B_0^{(15-12)}$	$B_0^{(11-8)}$	$B_0^{(7-4)}$	$B_0^{(3-0)}$	$H_0^{(3-0)}$	$H_{p0}^{(3-0)}$
1	$B_1^{(31-28)}$	$B_1^{(27-24)}$	$B_1^{(23-20)}$	$B_1^{(19-16)}$	$B_1^{(15-12)}$	$B_1^{(11-8)}$	$B_1^{(7-4)}$	$B_1^{(3-0)}$	$H_1^{(3-0)}$	$H_{p1}^{(3-0)}$
2	$B_2^{(31-28)}$	$B_2^{(27-24)}$	$B_2^{(23-20)}$	$B_2^{(19-16)}$	$B_2^{(15-12)}$	$B_2^{(11-8)}$	$B_2^{(7-4)}$	$B_2^{(3-0)}$	$H_2^{(3-0)}$	$H_{p2}^{(3-0)}$
3	$B_3^{(31-28)}$	$B_3^{(27-24)}$	$B_3^{(23-20)}$	$B_3^{(19-16)}$	$B_3^{(15-12)}$	$B_3^{(11-8)}$	$B_3^{(7-4)}$	$B_3^{(3-0)}$	$H_3^{(3-0)}$	$H_{p3}^{(3-0)}$
$\mathbf{V_{CC}}$	$V_{(31-28)}$	$V_{(27-24)}$	$V_{(23-20)}$	$V_{(19-16)}$	$V_{(15-12)}$	$V_{(11-8)}$	$V_{(7-4)}$	$V_{(3-0)}$		
$\mathbf{V_{PC}}$	$V_{p(31-28)}$	$V_{p(27-24)}$	$V_{p(23-20)}$	$V_{p(19-16)}$	$V_{p(15-12)}$	$V_{p(11-8)}$	$V_{p(7-4)}$	$V_{p(3-0)}$		

inter-word and intra-word MBUs in memory system. The differentiator of CMC from other coding techniques is soft error prediction, which predicts the exact number of soft errors present in the memories before the correction task.

2.1 Proposed CMC encoder and decoder

The cost of the ECC technique is directly proportional to the required redundant bits [11]. In the proposed CMC, group of words are taken as input to the encoder and decoder instead of single word taken in the existing works, for achieving lower redundant bits. i.e. N-bit words are arranged in M rows each forms a matrix of size M×N. Each word (row) is divided into k symbols of m bits N= k×m. The horizontal counter codes (H_{CC}), horizontal prediction codes (H_{PC}), vertical counter codes (V_{CC}) and vertical parity codes (V_{PC}) includes the vertical counter bits $V_{(3\text{-}0)\ldots}V_{(31\text{-}28)}$ and horizontal counter bits $H_0^{(3\text{-}0)}\ldots H_3^{(3\text{-}0)}$ for error prediction and the vertical parity bits $V_{P(3\text{-}0)\ldots}$ $V_{P\,(31\text{-}28),}$ horizontal parity bits $H_{P0}^{(3\text{-}0)}\ldots.$ $H_{P3}^{(3\text{-}0)}$ for error correction respectively. To explain the proposed CMC, 32-bit words are considered as an example, arranged in 4 rows each forms 4×32 matrix as shown in Table I. The required number of parity bits for the group length is given in Table II. It shows that more number of words in a group needs less number of redundant bits. For example the computation of redundant bits for 8 words in a group needs 64 redundant bits and 4 words in a two different group (2×48) is 96 redundant bits. But more number of words in a group will affect the percentage of correction coverage. For this reason this work limits the number of words in a group to 4.

Table 2: Required no. of parity bits per group

No. of words per group	No. Of Redundant bits
1	24
2	40
3	44
4	48
5	52
6	56
7	60
8	64

The proposed CMC has two steps, first combinational ones counter operation is performed on data bits for predicting and reducing the number of redundant bits for further error detection and correction. For an array of memory words, the horizontal (row) counter code bits can be calculated using Equation 1. For example the horizontal counter code of first row word is shown in (Equation 2) – (Equation 5)

$$H_M m = \sum_{k=0}^{k-1} B_M \,\hat{}\, (4k+m) \tag{1}$$

$$H_0^0 = B_0^0 + B_0^4 + B_0^8 + B_0^{12} + B_0^{16} + B_0^{20} + B_0^{24} + B_0^{28} \tag{2}$$
$$H_0^1 = B_0^1 + B_0^5 + B_0^9 + B_0^{13} + B_0^{17} + B_0^{21} + B_0^{25} + B_0^{29} \tag{3}$$
$$H_0^2 = B_0^2 + B_0^6 + B_0^{10} + B_0^{14} + B_0^{18} + B_0^{22} + B_0^{26} + B_0^{30} \tag{4}$$
$$H_0^3 = B_0^3 + B_0^7 + B_0^{11} + B_0^{15} + B_0^{19} + B_0^{23} + B_0^{27} + B_0^{31} \tag{5}$$

For an array of memory words, the vertical (column) counter code bits are calculated using Equation 6. For example the vertical counter code of first column is shown in (Equation 7)-(Equation 10)

$$V_N = \sum_{m=0}^{m-1} Bm\hat{}\,(N) \tag{6}$$

$$V_0 = B_0^0 + B_1^0 + B_2^0 + B_3^0 \tag{7}$$
$$V_1 = B_0^1 + B_1^1 + B_2^1 + B_3^1 \tag{8}$$
$$V_2 = B_0^2 + B_1^2 + B_2^2 + B_3^2 \tag{9}$$
$$V_3 = B_0^3 + B_1^3 + B_2^3 + B_3^3 \tag{10}$$

where k is the number of symbols in a word; m is the number of bits in a symbol and M is the number of words in the array. In the second step horizontal and vertical parity bits can be calculated from the horizontal and vertical counter codes. Horizontal parity bits are calculated from horizontal counter codes using Equation 11. Similarly, vertical parity bits are calculated from horizontal counter codes using Equation 12. Finally, both the intra and inter word errors will be corrected in decoding step.

$$H_{pM} m = \{\ 0\ \ for\ \ H_M m = 0,2,\ldots k\ ;\ 1\ \ for\, H_M m = 1,3,\ldots k-1\ ; \tag{11}$$

$$V_{pN} m = \{\ 0\ \ for\ \ V_N m = 0,2,\ldots k\ ;\ 1\ \ for\, V_N m = 1,3,\ldots k-1\ ; \tag{12}$$

The encoding and decoding algorithms are given below to understand the flow.

Algorithm for Encoding.

ACW - Array of configuration word
H_{CC} – Horizontal (row) counter codes
V_{CC} – Vertical (column) counter codes
H_{PC} – Horizontal (row) parity codes
V_{CC} – Vertical (column) parity codes

Input: *ACW [4 ×32=128 bits]*
Output: H_{CC}, V_{CC}, H_{PC}, V_{PC}
1: *ACW to be written*
2: Split into K symbols per Configuration word
2: while *symbols* = *true* do
3: for all $H_N m \in H_{CC}$ do *onescount* (*ACW*);
4: for all $H_{pm} m \in H_{PC}$ do *parity*(H_{CC});
5: for all $V_N \in V_{CC}$ do *onescount* (*ACW*);
6: for all $V_{pN} m \in V_{PC}$ do *parity*(V_{CC});
6: update H_{CC}, V_{CC}, H_{PC}, *and* V_{PC};
7: end while
8: return *ACW*;

Algorithm for *Decoding*.

Input : Errored *ACW[4 ×32=128 bits], Hcc, Vcc, Hpc, Vpc*
Output : error prediction value (epv) , corrected word (cw)
1: Read *errored ACW [ACWm]*
2: Split into K symbols per Configuration word
3: Read Hcc,Vcc,Hpc,Vpc
4: while *symbols* = *true* do
5: for all $H_N m \in Hccm$ do *onescount* (*ACWm*);
6: for all $H_{pm} m \in Hpc$ do *parity*(*Hcc'*);
7: for all $V_N \in Vccm$ do *onescount* (*ACWm*);
8: for all $V_{pN} m \in Vpc$ do *parity*(*Vcc'*);
9: update *Hcc',Vcc',Hpc',Vpc'*;
10: find hsc= diff(Hcc-Hcc')
11: find hsp= diff(Hpc-Hpc')
12: find vsc= diff(Vcc-Vcc')
13: find vsp= diff(Vpc-Vpc')
14: if((hsp==0)&(vsp==0))
15: begin
16: {Syndrome =0
17: error=0 }
18: end
19: else
20: begin
21: { Syndrome ≠ 0
22: error ≠ 0
23: epv = {hsc,vsc};
24: $B_{intracorrect}$ = ACW m XOR vs
$B_{intercorrect}$ = ACW m XOR Hs } }
25: end
26: end while
22: return *ACW*;

2.2 Proposed fault-tolerant memory architecture

The proposed fault-tolerant memory architecture is illustrated in Figure 1. First, for the period of encoding process, original data bits *D* are fed to the encoder, and then H_{CC}, H_{PC} and V_{PC} are obtained from the CMC encoder. The obtained CMC codeword consist data and redundancy bits, which are stored in the separate SRAM memories. The MBUs occurred in the memory is being corrected at the decoding process using the CMC Encode-Compare.

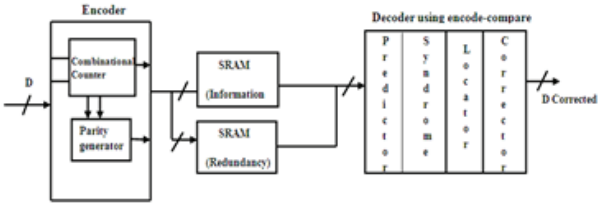


Figure 1: Fault-tolerant memory architecture

The detail architecture of CMC encoder is shown in Figure 2. First, the H_{CC} *and* V_{CC} bits are computed by performing 8-bit combinational counting operation of selected sliced bits of symbols per row and 4-bit combinational counting operation of selected sliced bits of symbols per column respectively. Second the 4-bit H_{PC} are computed by performing XOR operations of respective row H_{CCs}, totally 16 bit H_{PCs} are computed for 4 rows. The 1-bit V_{PC} is computed by performing XOR operations of respective column VCCs, totally 32 bit VPCs are computed for 32 columns.

The proposed CMC Encoder consists of two combinational ones counter circuits, namely 8-bit combinational ones counter and 4-bit combinational ones counter. The 8-bit combinational ones counter (Row counter) shown in Figure 3(a). The row counter counts the number of one's using 9 half adders (HAs), 2 full adders (FAs) and 2 XOR gates and is given in (Equation 13). Similarly, the 4-bit combinational ones counter (Column counter) shown in Figure 3(b) counts the number of one's using 4 half adders (HAs), and one XOR gate and is given in (Equation 14). The detail architecture of CMC decoder is shown in Figure 4. Decoder consists of predictor, syndrome calculator (detector), locator and corrector. Horizontal and vertical syndrome calculator are used to detect and locate the MBUs in the memories.

$$\left.\begin{aligned} out[0] &= (a\oplus b)\oplus(c\oplus d)\oplus(e\oplus f)\oplus(g\oplus h)\\ out[1] &= (a\oplus b).(c\oplus d)\oplus(a.b)\oplus(c.d)\oplus(e\oplus f)(g\oplus h).(e.f)(g.h)(a\oplus b)\oplus(c\oplus d).(e\oplus f)\oplus(g\oplus h)\\ out[2] &= (a.b.c.d)(e.f.g.h)+[(a.b.c.d)\oplus(e.f.g.h)][(a\oplus b).(c\oplus d)\oplus(a.b)\oplus(c.d).(e\oplus f).(g\oplus h)\oplus(g.h)(e.f)\\ out[3] &= a.b.c.d.e.f.g.h \end{aligned}\right\} \quad (13)$$

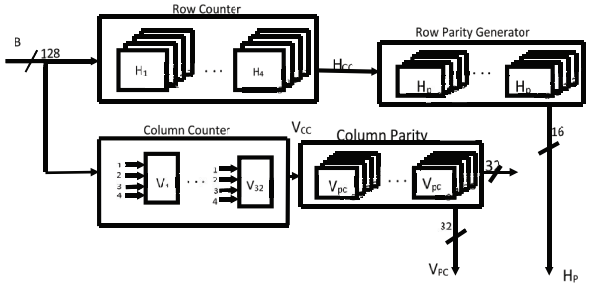
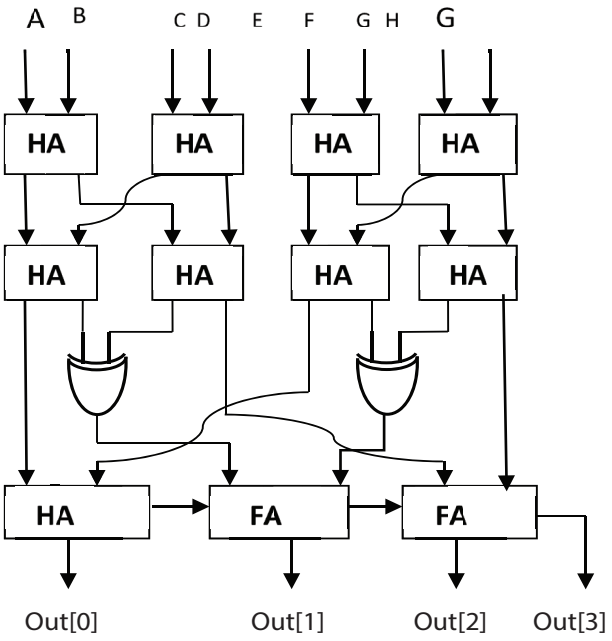
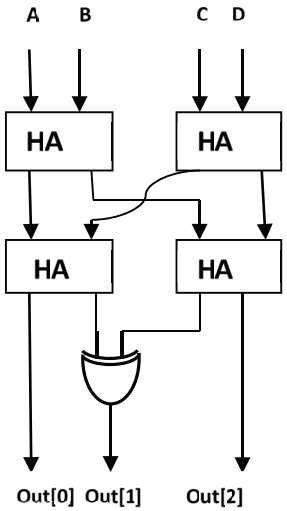


Figure 2: Architecture for CMC Encoder.



(a) Row Counter



(b) Column Counter

Figure 3: 1's counters (a) Row counter (b) Column Counter.

$$\left.\begin{aligned} out[0] &= a \oplus b \oplus c \oplus d \\ out[1] &= (a \oplus b).(c \oplus d). \oplus (a.b). \oplus (c.d) \\ out[2] &= a.b.c.d \end{aligned}\right\} \quad (14)$$

Finally corrector is used to correct the erroneous bits based on horizontal syndrome, vertical syndrome and erroneous bits.

The following example gives the computation of horizontal, vertical parity bits for MBU detection and correction for a group of words. Let us consider the original information bits (B) as 128 bits. It can be divided into four rows, each containing 32 bits. Each row is divided into 8 symbols, each containing four bits. H_{CC} and V_{CC} are horizontal ones counter (Row counter) and Vertical ones counter (Column counter) for predicting soft errors and reducing the number of redundant bits. An H_{PC} and V_{PC} bit detects and corrects the errors in 128-bits. For example the original 128-bits information is shown in Table III (a), may have intra-word errors as shown in Table III (b), and inter-word errors are shown in Table III (c), for 128-bits information. The horizontal counter codes were calculated using Equation (1)-(5) and vertical counter codes were calculated using Equation (6)-(10). The horizontal and vertical parity bits were calculated using Equation (11) and Equation (12) respectively. Finally, both the intra and inter word MBUs can be corrected by the decoding algorithm.

3 Correction coverage and overhead analysis

In this section, the proposed CMC has been coded in Verilog hardware description language (HDL), simulated using Xilinx-Isim and tested its functionality for various inputs. The correction coverage and overhead analysis have been done. For fair comparisons, Hamming [8] [9], MC [11], DMC [12], SHMC [14], RMC [14], I3D [22], XILINX CRC [19] [20] are used for reference.

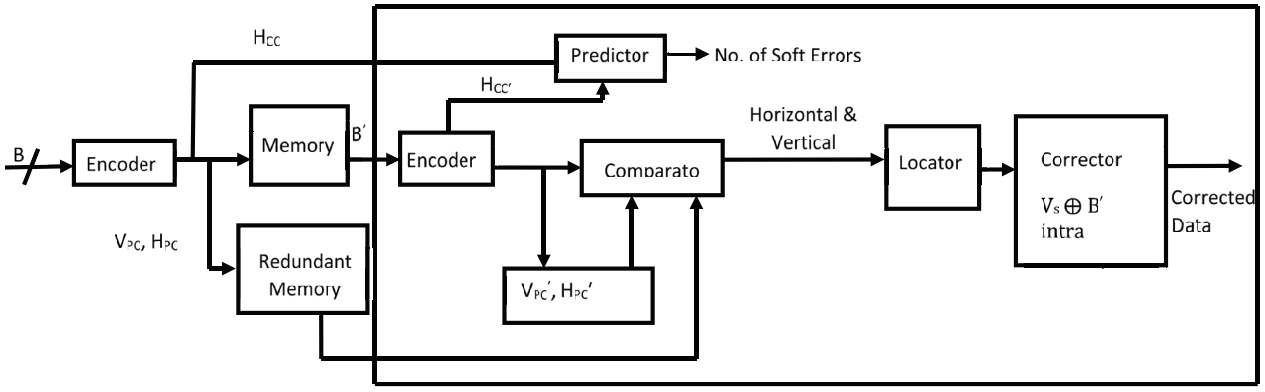


Figure 4: Architecture for CMC Decoder.

Table 3: (a) 128-bit logical organization of cmc

S.NO	Symbol8	Symbol7	Symbol6	Symbol5	Symbol4	Symbol3	Symbol2	Symbol1	H_{CC}	H_{PC}
1	1010	1010	1010	1010	1010	1010	1010	1010	8080	0000
2	0101	0101	0101	0101	0101	0101	0101	0101	0808	0000
3	1001	1001	1001	1001	1001	1001	1001	1001	8008	0000
4	0110	0110	0110	0110	0110	0110	0110	0110	0880	0000
V_{CC}	2222	2222	2222	2222	2222	2222	2222	2222		
V_{PC}	0000	0000	0000	0000	0000	0000	0000	0000		

(b) Intra word error version

S.NO	Symbol8	Symbol7	Symbol6	Symbol5	Symbol4	Symbol3	Symbol2	Symbol1	H'_{CC}	H'_{PC}
1	0101	0101	0101	0101	0101	0101	0101	0101	0808	0000
2	0101	0101	0101	0101	0101	0101	0101	0101	0808	0000
3	1001	1001	1001	1001	1001	1001	1001	1001	8008	0000
4	0110	0110	0110	0110	0110	0110	0110	0110	0880	0000
V'_{CC}	1313	1313	1313	1313	1313	1313	1313	1313		
V'_{PC}	1111	1111	1111	1111	1111	1111	1111	1111		

(C) Inter word error version

S.NO	Symbol8	Symbol7	Symbol6	Symbol5	Symbol4	Symbol3	Symbol2	Symbol1	H'_{cc}	H'_{pc}
1	0101	1010	1010	1010	1010	1010	1010	1010	7171	1111
2	1010	0101	0101	0101	0101	0101	0101	0101	1717	1111
3	0110	1001	1001	1001	1001	1001	1001	1001	7117	1111
4	1001	0110	0110	0110	0110	0110	0110	0110	1771	1111
V'_{cc}	2222	2222	2222	2222	2222	2222	2222	2222		
V'_{pc}	0000	0000	0000	0000	0000	0000	0000	0000		

3.1 MBU Patterns

In 2009, E. Ibe et .al analyzed the scaling effects on neutron induced soft error in SRAM array down to 22 nm technology node and they observed that nearly 50 % of soft errors are MBU incidents [21]. In order to fairly enumerate the MBU correction coverage of the proposed CMC technique, the detailed information about the possible MBU error patterns of 28nm SRAM array and their individual occurrence probabilities are needed. Figure 5 shows the MBU patterns and their occurrence probabilities [22] –[23].

3.2 Comparison for correction coverage

To facilitate the benefits and drawbacks of the proposed scheme, it is extensively compared with previous techniques. Simulation based MBU injection experiment has been done to extract error correction coverage of the previous techniques. The original 128-bit information and the faulty information can be specified in the text fixture, and fault injection can be implemented in a test-bench. Both single and multiple bit faults were injected, in case of MBU injection around one million combinations were injected. The correction coverage of various MBU mitigation techniques such as CMC, DMC, MC, and Hamming is obtained for various intra-word error test cases and it is shown in Figure 6. It is clear that the DMC performs 100% intra error correction up to 5 bit errors and 11.8% error correction in 16 bit errors. Similarly, MC performs 100% intra error correction up to 2 bits and 0.6% error correction up to 8 bits. But the proposed CMC provides 100% protection that is possible error correction up to 32 bits. In addition to that the correction coverage depicted in Table V compares the proposed technique, proven soft error mitigation techniques and existing research techniques.

The possibility of correction coverage is tested for larger the word widths which results the higher the correction capabilities. The maximum correction capability (MCC) is given in Table IV. In DMC, the correction capability for a 64- bit and 128-bit word is up to 9 bits and 17 bits respectively. In proposed CMC, the correction capability for a 64-bit and 128-bit word is up to 36 bits

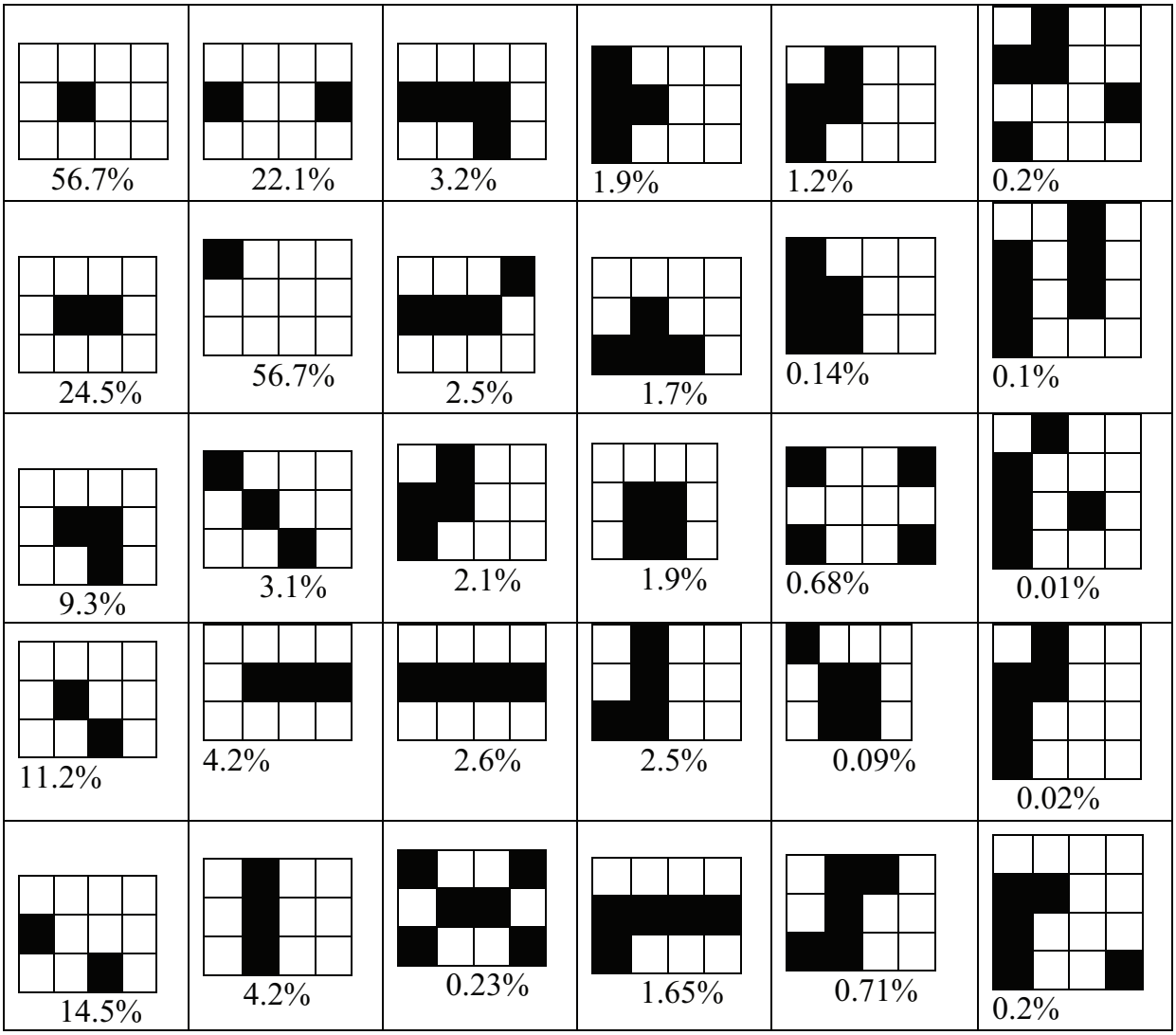


Figure 5: MBU patterns of high occurrence probabilities in 28nm SRAM array [22]-[23]

and 44 bits respectively. The results depicted in Table IV show that proposed CMC exceeds the performance of other codes by its efficient error tolerance capability against larger the MBU widths.

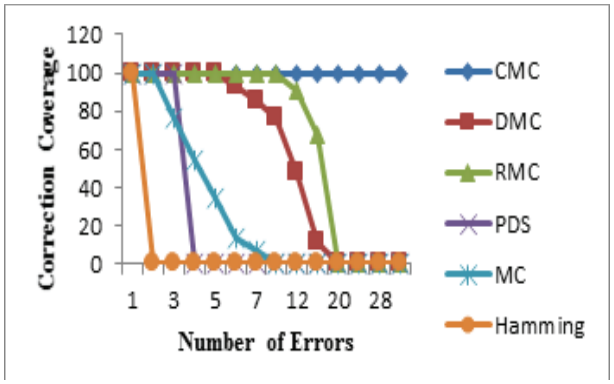


Figure 6: Intra word Correction coverage for various ECCs

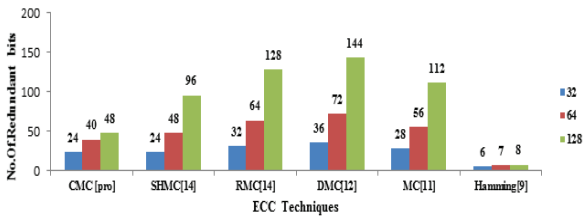


Figure 7: Required number of redundant bits for various error correction codes

Table 4: Maximum Correction Capability (MCC)

Technique	MCC (64-bits)	MCC (128-bits)
CMC	36 bits	44 bits
RMC	16 bits	32 bits
DMC	9 bits	17 bits
MC	4 bits	8 bits

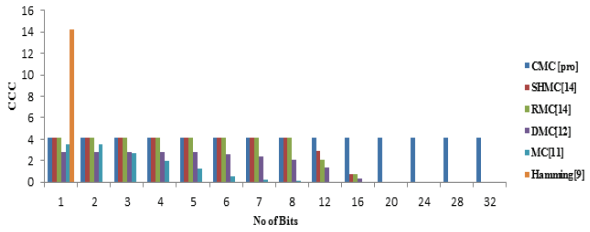


Figure 8: Correction coverage per cost for various error correction codes

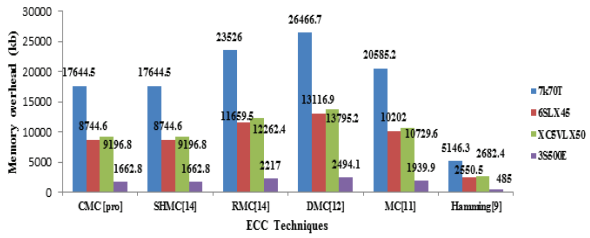


Figure 9: Intra word Memory Area overhead analysis of various Xilinx FPGA Devices.

3.3 Comparison for overhead analysis

In order to evaluate the efficiency of error mitigation techniques, the implementation overheads of these protection codes have to be analyzed. This paper analyzes the overheads in terms of cost and correction coverage per cost (CCC). The term cost indicates the number of redundant bits required to implement the error correction codes [11]. The cost for the proposed and typical coding techniques is portrayed for 32, 64 and 128 bits in Figure 7. This implies Hamming code needs very less number of redundant bits, but their correction capability is limited to 1. DMC need more number of redundant bits compared to all other codes. Linear increasing of redundant bits for the higher word lengths of the traditional codes were shown in Figure 7. The proposed CMC needs less number of redundant bits compared to all other codes due to the inter word processing capability. The CCC results of the proposed and typical coding techniques are portrayed up to 32 bits in a word is shown in Figure 8. This implies that coding techniques should have high value of the CCC for higher reliable solution. It should be noticed that when the number of errors is more than one per word, Hamming code cannot correct any errors. The proposed CMC provides consistent performance compared to all typical coding techniques. Thus, based on the analysis given in Figure 7 and Figure 8, the proposed CMC technique is better suited for low cost and safety critical (high- Performance) applications.

The best metric used to select the appropriate coding technique for the practical solutions is mean time to repair (MTTR) which is analyzed for all soft error mitigation techniques and portrayed in Table V. MTTR-R represents the actual MTTR and additional recovery time. The results shown in the Table V implies that proven mitigation techniques [19], Xilinx CRC+ECC [20] needs minimum MTTR value, but the correction coverage for the recent scaled technology (28 nm) is not satisfactory. The technique presented in the Xilinx CRC+Reload [20] gives 100% correction coverage, but they require MTTR as almost 3-times of the other techniques and this MTTR overhead is not acceptable in real time. Next the coding techniques presented in the [14] require minimum MTTR due to Encode-Compare mechanism,

Table 4: Comparison of different soft error mitigation techniques

Soft Error Correction Techniques	MTTR (ms)	MTTR-R (ms)	Correction coverage (%)	Distinguished Note
Proven Mitigation Techniques				
Xilinx SEU Correction [19]	9.342	0	51.72	Single bit correction
Xilinx CRC+ECC [20]	9.342	0	61.1	Global detection & Single bit correction
Xilinx CRC +Reload [20]	9.342	18.7	100	External Storage required
Existing Research Techniques				
Hamming code[8],[9]	10.7	0	51.652	Decode-Compare
DMC [12]	9.6	0	95.823	Decode-Compare
MC [11]	11.2	0	93.81	Decode-Compare
SHMC [14]	6.57	0	95.913	Encode-Compare
RMC[14]	6.68	0	94.62	Encode-Compare
I3D[22]	9.343	0.351	94.2	Erasure code
Proposed Technique				
CMC[Pro]	9.387	0	100	Prediction & Encode-Compare

but the correction coverage is not a maximum. DMC technique requires 9.6 ms for correcting the errors and the respected correction coverage is only about 95.823% [12]. The recent technique I3D requires 9.343 ms for detecting the error and 0.351 ms for recover the particular error word, the total MTTR is 9.694 ms and the respected correction coverage is only about 94.2% [22]. The proposed CMC require only 9.387 ms for correcting all error patterns shown in the Figure 7 and this MTTR value is almost equivalent to the proven techniques .Thus the proposed CMC technique can be used in safety critical applications compared to all typical coding techniques. Finally memory overhead for storing the redundant bits in Xilinx FPGA devices are shown in Figure 9. This implies that Hamming code need minimum memory overhead but the correction capability is limited to 1. The proposed CMC and the SHMC technique presented in [14] are require acceptable level of redundant memory overhead compared to all other codes.

4 Conclusion

In this paper, a novel technique CMC is proposed to cope with radiation induced MBUs. The obtained results showed that the proposed scheme has a better protection level against huge MBUs in the intra and inter words of the memory. The proposed CMC utilized Encode-Compare mechanism to predict and correct errors for a group of words, so that the MTTR value is minimum and equivalent to proven mitigation techniques with improved correction coverage. The only drawback of the proposed work is the requirement of more redundant bits to protect memory. In future the research will be conducted for improving reliability and reducing cost of the proposed technique for the below 28 nm FPGAs.

5 Acknowledgements

This work was supported in part by the University Grant Commission (UGC), Government Of India, National Fellowship under Grant NFO25109.

6 References

1. C. Argyrides, C. Lisboa, L. Carro and D.K. Pradhan, " A soft error robust and power aware memory design " in Proc. 20th Annu, Symp, Integr, Circuits Syst Des (SBCCI), Sep.2007, pp.300–305. www.inf.ufrgs.br/~calisboa/.../SlidesSBCCI2007ETLPRAM.pdf
2. M.J. Wirthlin, "FPGAs Operating In A Radiation Environment: Lessons Learned From FPGA In Space," workshop on electronics for particle physics, Oxford, U.K , September 2012, pp. 17–21. https://indico.cern.ch/event/.../twepp_wirthlin_Sept_2012.ppt.pdf
3. Xilinx, "Device Reliability Report, UG116, v10.1", August. 2014. juhuj.com/open-file-pdf-convert-pdf-download-ug116.htm
4. D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device," *IEEE Trans.Nucl. Sci.*, vol. 52, no. 6, pp. 2433–2437, Dec. 2005. ieeexplore.ieee.org/document/1589220/
5. R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel.*, vol. 5, no.3, pp. 305–316, Sep. 2005. ieeexplore.ieee.org/document/1545891/
6. ITRS 2002. [Online]. Available: http://public.itrs.net
7. P. M. B. Rao, M. Ebrahimi, R. Seyyedi, and M. B. Tahoori, "Protecting SRAM-based FPGAs against multiple bit upsets using erasure codes," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6. http://ieeexplore.ieee.org/document/6881539/?reload=true&arnumber=6881539
8. A. Sanchez-Macian, P. Reviriego, 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,pp.574-576,March2014. http://ieeexplore.ieee.org/document/6217302/
9. D. Houghton, "The Engineer's Error Coding Handbook". Chapman and Hall, London, U.K , 1997. www.springer.com/gp/book/9780412790706
10. 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. ieeexplore.ieee.org/document/6026914/
11. 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. ieeexplore.ieee.org/document/5352255/
12. Jing Guo, Liyi Xiao, Zhigang Mao, Qiang Zhao, "Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol.22, no.1, pp.127-135, Jan. 2014. http://ieeexplore.ieee.org/document/6487418/
13. 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. www.isi.edu/~draper/papers/esscirc08.pdf

14. A. Ahilan, P. Deepa, "Design for Built-In FPGA Reliability via Fine-Grained 2-D Error Correction Codes", *Microelectronics Reliability*, vol. 55, pp. 2108-2112, Aug. –Sep. 2015. http://www.sciencedirect.com/science/article/pii/S0026271415001675?np=y
15. 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. https://www.lume.ufrgs.br/bitstream/handle/10183/27598/000459042.pdf?sequence=1
16. 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. http://iosrjournals.org/iosr-jece/papers/Vol8-Issue2/M0827178.pdf
17. Appathurai, A.; Deepa, P., "Design for reliablity: A novel counter matrix code for FPGA based quality applications," in Proc. *6th Asia Symposium on Quality Electronic Design (ASQED), Aug. 2015*, pp.56-61. http://ieeexplore.ieee.org/document/7274007/
18. L. Jones, "Single event upset (SEU) detection and correction using Virtex-4 devices," Xilinx Corporation, San Jose, CA, USA, Appl. Note XAPP714, 2007. http://www.eng.auburn.edu/~strouce/class/bist/CATA09seu.pdf
19. Xilinx, "LogiCORE IP soft error mitigation controller, PG036, v3.4" San Jose, CA, USA, 2012. www.xilinx.com/support/documentation/ip.../v3_4/pg036_sem.pdf
20. 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. http://ieeexplore.ieee.org/document/5467170/
21. E. Costenaro, D. Alexandrescu, K. Belhaddad, and M. Nicolaidis, "A practical approach to single event transient analysis for highly complex design," *J. Electron. Test.*, vol. 29, no. 3, pp. 301–315, 2013. http://ieeexplore.ieee.org/document/6104439/
22. M. Ebrahimi, P.M.B. Rao,; R. Seyyedi,; M.B . Tahoori, "Low-Cost Multiple Bit Upset Correction in SRAM-Based FPGA Configuration Frames," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012. http://ieeexplore.ieee.org/document/7104165/
23. *JEDEC89C Standard*, [Online]. Available: http://www.jedec.org/standards-documents, accessed Apr. 2015.

Arrived: 26. 09. 2016
Accepted: 13. 12. 2016