ISSN XXXX XXXX © 2019 IJESC

Research Article Volume 9 Issue No.3

Counter Matrix Error Correction Code for Memory Based Application

V. Monisha[1], R. Shinoji[2]
Assistant Professor[1], ME VLSI Student[2]
Department of Electronics and Communication Engineering
Dr.Sivanthi Aditanar College of Engineering, India

Abstract:
Error Correction Codes (ECCs) are elementary in many applications for regenerating received data or guaranteeing accuracy of information in a memory system. The information data is encoded and stored in memory. It may be affected which leads to error. To eradicate this problem soft error mitigation techniques such as hamming codes, symbolic hamming codes (Flexible Unequal ECC) are used earlier to correct one bit and 4 bit error. To correct the error with minimum number of redundant bits, a Counter Matrix Code is proposed which corrects the bits upto 32 bit error for 64 bit input.

Keywords: Error correction codes [ECC], fault tolerance, multiple cell upsets, Hamming codes.

1. INTRODUCTION

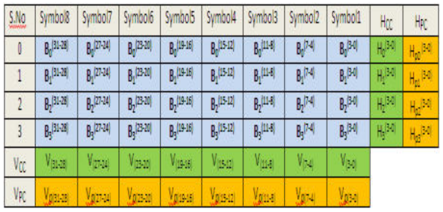
Error coding is a method of detecting and correcting the errors to ensure information is transferred intact from its source to its destination. Error correction coding uses mathematical formulae to encode data bits at the source into longer bit words for transmission. The "code word" can then be decoded at the destination to retrieve the information. The extra bits in the code word provide redundancy that the coding scheme used, will allow the destination to use the decoding process to determine if the communication medium introduces errors and in some cases correct them so that the data need not be retransmitted. Hamming code is a set of error-correction code s that can be used to detect and correct bit errors that can occur when computer data is moved or stored. Hamming code is named for R. W. Hamming of bell labs. When it comes to storage, redundancy can be a safeguard or take the form of unwanted overhead. An error-correcting code (ECC) or forward error correction (FEC) code is a process of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission. Error correction is the process of detecting errors in transmitted messages and reconstructing the original error-free data. Error correction ensures that corrected and error-free messages are obtained at the receiver side. Forward error correction (FEC) is a method that involves adding parity data bits to the message.These parity bits will be read by the receiver to determine whether an error happened during transmission .In this case, the receiver checks and corrects errors when they occur. It does not ask the transmitter to resend the frame or message. Error correction is the process of detecting errors in transmitted messages and reconstructing the original error-free data. Error correction ensures that corrected and error-free messages are obtained at the receiver side. Forward error correction (FEC) is a method that involves adding parity data bits to the message. These parity bits will be read by the receiver to determine whether an error happened during transmission or storage. In this case, the receiver checks and corrects errors when they occur. It does not ask the transmitter to resend the frame or message. Error detection refers to the techniques used to detect noise or other impairments introduced into data while it is transmitted from source to destination. Error detection ensures reliable delivery of data across vulnerable networks. Error detection minimizes the probability of passing incorrect frames to the destination, known as undetected error probability. A code with minimum hamming distance, d, can detect up to d − 1 errors in a code word. Codes with minimum hamming distance d = 2 are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code. To detect and correct the errors, additional bits are added to the data bits at the time of transmission**.** The additional bits are called parity bits. They allow detection or correction of the errors. The data bits along with the parity bits form a code word.

2. COUNTER MATRIX CODE

In this section, CMC encoding and decoding algorithm is proposed to predict and correct the MBUs, and the FPGA architectures for encoder and decoder are presented. The proposed CMC based encoding and decoding algorithm appears to lend itself to detect both inter-word and intra-word MBUs in the memory system. The soft error prediction and the exact number of soft errors present in the memories has been known before the correction task in CMC than other coding techniques.

A. CMC ENCODER AND DECODER

In the proposed CMC, a group of N-bit words are arranged in M rows each forming a matrix of size $M \times N$. Each word (row) is divided into k symbols of m bits , where $N = k \times m$. The horizontal counter codes (H_{CC}), horizontal prediction codes (H_{PC}), vertical counter codes (V_{CC}), and vertical parity codes (V_{PC}) include the vertical counter bits $V_{(3-0)}$ … $V_{(31-28)}$ and horizontal counter bits $H_0^{(3-0)}$ … $H_3^{(3-0)}$ for error prediction, and the vertical parity bits $V_{P(3-0)}$ …$V_{P\ (31-28)}$, horizontal parity bits $H_{P0}^{(3-0)}$ … $H_{P3}^{(3-0)}$ for error correction. To explain the proposed CMC, 32-bit words are considered as an example, arranged in 4 rows each forming 4×32 matrix, as shown in Figure1.



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	$B_3^{[31-28]}$	$B_3^{[27-24]}$	$B_3^{[23-20]}$	$B_3^{[19-16]}$	$B_3^{[15-12]}$	$B_3^{[11-8]}$	$B_3^{[7-4]}$	$B_3^{[3-0]}$	$H_3^{[3-0]}$	$H_{p3}^{[3-0]}$
V_{CC}	$V_{[31-28]}$	$V_{[27-24]}$	$V_{[23-20]}$	$V_{[19-16]}$	$V_{[15-12]}$	$V_{[11-8]}$	$V_{[7-4]}$	$V_{[3-0]}$		
V_{PC}	$V_{p[31-28]}$	$V_{p[27-24]}$	$V_{p[23-20]}$	$V_{p[19-16]}$	$V_{p[15-12]}$	$V_{p[11-8]}$	$V_{p[7-4]}$	$V_{p[3-0]}$		

Figure .1. 128-bit logical organization of CMCs

The required number of parity bits for the group length is given in Table 1It shows that more number of words in a group needs a 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 be 4.

Table.1 Required Number of Parity Bits per Group

No.of Word per Group	No. of Redundant Bits
1	24
2	40
3	44
4	48
5	52
6	56
7	60
8	64

B.FAULT-TOLERANT MEMORY ARCHITECTURE

The proposed fault-tolerant memory architecture is illustrated in Figure2. 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 of data and redundancy bits, which are stored in separate SRAMs. The MBUs occurred in memory is being corrected at decoding process using CMC encode–compare function.

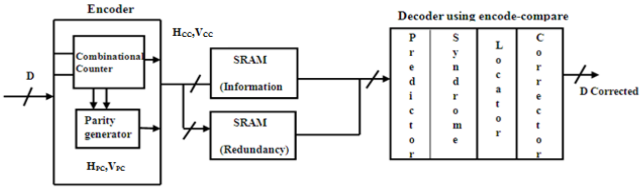


Figure.1.A Fault-Tolerant Memory Architecture

First, the H_{CC} and V_{CC} bits are computed by performing 8-bit combinational counting operation of the selected sliced bits of symbols per row and 4-bit combinational counting operation of the selected sliced bits of symbols per column.

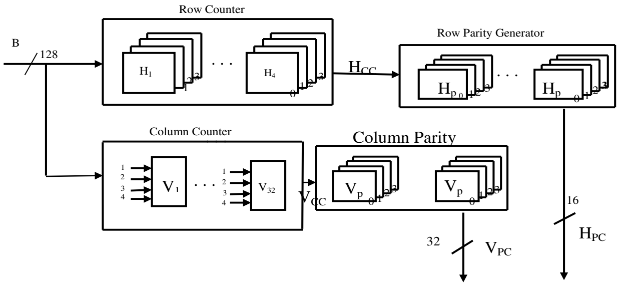


Figure.3 Architecture of a CMC Encoder

Second, the 4-bit H_{PC} are computed by performing X-OR operations of the respective row H_{CCs}; totally 16-bit H_{PCs} are computed for four rows. The 1-bit V_{PC} is computed by performing X- OR operations of respective column VCCs; totally 32-bit V_{PC}s are computed for 32 columns. The fundamental function of the combinational ones counter is to count the number of 1s along with its input bits. The proposed CMC encoder consists of two combinational ones counter circuits, namely 8-bit and 4-bit combinational ones counter. The row counter counts the number of one's using nine half adders (HAs), two full adders (FAs), and two XOR gates and is given in Equation 1 .Similarly, the 4-bit combinational ones counter counts the number of one's using four half adders (has) and one XOR gate, and is given in Equation 2.

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

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

The encode–compare mechanism is used instead of actual decode–compare mechanism in the decoder for fast error correction. The decoder consists of a predictor, syndrome calculator (detector), a locator, and a corrector. The predictor compares H_{CC} and $H_{CC}{}'$ and identifies the number of soft errors present in the memories. The horizontal and vertical syndrome calculator is used to detect and locate the MBUs in memories. Finally the corrector is used to correct the erroneous bits based on horizontal syndrome, vertical syndrome, and erroneous bits.

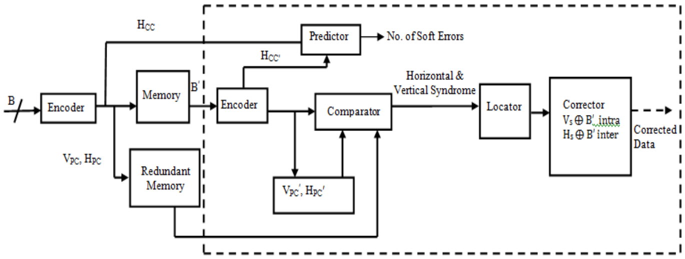


Figure.4. Architecture for CMC Decoder

3. EXPERIMENTAL RESULTS AND DISCUSSION

This paper presents the analysis of Counter Matrix Code (CMC) is discussed. Encoder and decoder architectures are used for finding the redundancy bits and syndrome to detect and correct the error. This technique can correct the error up to 32 bits for 64 bit input. Less redundant bits are used here so the area and the power consumption is low. The software tools used for the analysis Xilinx 14.5.The language used in the simulation process is Verilog, which is the hardware description language. The result for this novel redundant code method is discussed in detail

3.1. RTL Schematic

Register transfer level (RTL) is a design abstraction which models a circuit in terms of the flow of the original data, erroneous data and the corrected data. In Counter Matrix Code (CMC) there are two inputs di(63:0),do(31:0).The NRC method outputsarecrct_ data(63:0).

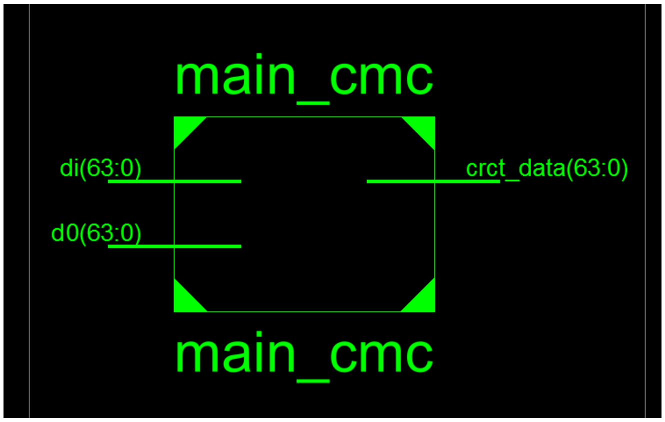
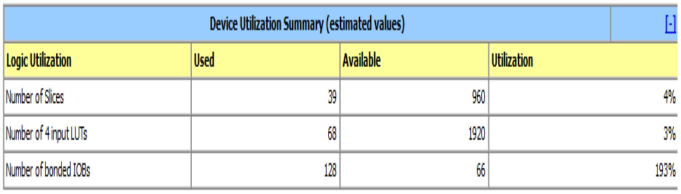


Figure .3.1.RTL Schematic for CMC

3.2. Resource utilization

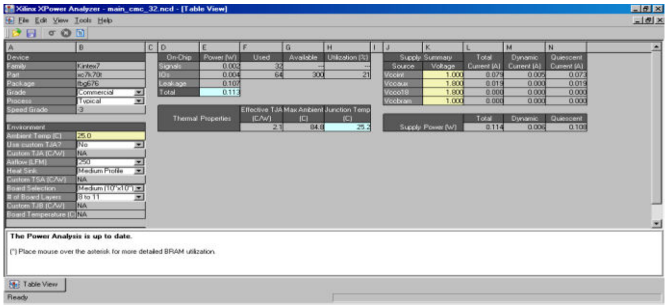
In CMC device utilization summary represents the number of devices. The number of slice of LUT used is 39 and the number of bonded IOBs is 128.



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	39	960	4%
Number of 4 input LUTs	68	1920	3%
Number of bonded IOBs	128	66	193%

Figure 3.2 Device utilization summary for CMC Method.

3.3 Power analyser for CMC method


Xilinx XPower Analyzer - main_cmc_32.ncd - [Table View]
File Edit View Tools Help
Table View
Ready

Device	
Family	Kintex7
Part	xc7k70t
Package	fbg676
Grade	Commercial
Process	Typical
Speed Grade	-3
Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Signals	0.002	32	---	---
IOs	0.004	64	300	21
Leakage	0.107			
Total	0.113			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.1	84.8	25.2

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.079	0.005	0.073
Vccaux	1.800	0.019	0.000	0.019
Vcco18	1.800	0.000	0.000	0.000
Vccbram	1.000	0.000	0.000	0.000

	Total	Dynamic	Quiescent
Supply Power (W)	0.114	0.006	0.108

The Power Analysis is up to date.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

Figure.3.3. Power consumption for the CMC method is 0.114watts.

4. REFERENCES

[1]. Altaf Mukati, "A survey of memory error correcting techniques for improved reliability" in ELSEIVER, Journal of Network and Computer Applications 34 (2011).

[2]. Costas Argyrides, Dhiraj .K, Pradhan, Taskin Kocak, Matrix code for reliable and cost efficient memory chips", IEEE transactions on very large scale integration (vlsi) systems, vol. 19, no. 3, march 2011

[3]. Eishi ibe, yasuo yahagi, ken-ichi shimbo, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22nm design rule", IEEE transactions on electron devices, vol.57, no.7, july 2010.

[4]. Felipe silva, otavia,cesar macron, "evalution of multiple bit upset tolerant codes for NoCs buffering", IEEE 2017.

[5]. Hamming R.W, "error detecting and correcting codes", Bell. syst.Tech.j., vol 29,no 2,apr 2015.

[6]. Helano de S. Castro, Jarbas A. N. da Silveira, "A correction code for multiple upsets in memory devices for space applications", in proc.14th IEEE Int... New Circuits Syst. Conf. (NEWCAS), jun.2016

[7]. Joaquin Gracia –Moran, Luis J.Saiz Adalid, Daniel Gil-Tomas "Improving Error Correction Code for multiple upsets in space application" IEEE transaction on very large scale integration(VLSI) system,2018

[8]. Luis-J. Saiz-Adalid, Pedro Gil, "Modified hamming code to enhance short burst error detection in semiconductor memories", Tenth European Dependable Computing Conference, 2014

[9]. Matthew j. Gadlange, Austin H.Roach, "Multiple-cell upsets induced by single high-energy.