An Approach to Detect and Correct Single Bit Data Error Using Reed_Muller Matrix

Monika Gope, Md. Ariful Islam Khandaker, Mehnuma Tabassum Omar

Department of Computer Science and Engineering
Khulna University of Engineering & Technology (KUET)
Khulna 9203, Bangladesh.

Abstract: Transferring data between two points is very crucial, also for some vital applications the precision of the transferred data is extremely important, however an error throughout the transmission of data is awfully familiar [1]. Error correcting codes are very valuable in transfer information over lengthy distances or through channels where errors might take place in the message. They have become more common as telecommunications have expanded and developed a use for codes that can auto-correct. Reed–Muller codes are a family of linear error correcting codes used in communications [2].For bit error correction, the Reed-Muller code is one of the efficient algorithms, on the other hand, its high-computation prerequisite innate in the decoding process interdict its use in handy applications [3].This paper describes a new method to detect and correct a single bit in the data message using Reed Muller matrix. The aim is to provide error-free information through transmitting and receiving by detection and correction a single bit in a big data very efficiently. The design detects and corrects all single bit errors in a 16, 32, 64, 128 bit data, and used 4, 5, 6, 7 check bits respectively. In the proposed method, it is feasible to detect the precise place of single bit error and correct that using least check bits.

Index Terms- Error Correction and detection, Reed Muller, Coding, Single bit error, Check bit

I. INTRODUCTION

In the present world, communication has got numerous applications such as telephonic conversations in which the messages are encoded into the communication channel and then decoding it at the receiver end [4]. Through data transmission, the arbitrary bit errors can be formed by ecological obstruction and material defects in the communication medium. Error coding is a technique of detecting and correcting these errors to make sure data is transferred unbroken from its source to its target. Computing in computer memory, magnetic and optical data storage technology, satellite, space and network communications, mobile networks, and almost any other form of digital data communication the error coding is used for fault tolerant. For communication error coding uses arithmetical formulas to encode data bits at the source into longer bit words and at the destination it decode the received bit words with the decoding method and determine the original massage. The earlier approach uses Error-Correcting Codes and latter uses Error-detecting Codes [5]. A method is basically worthless if it does not assure that the data received by one device are the same as the data transmitted by a different device. The main objective of this study is to device a coding scheme which is able to detect and correct such errors [6].

The rest of the paper is prepared as follows. In section 2, we briefly describe the highlights of the background of the coding scheme used in this paper. Section 3 presents our algorithm for matrix generation and check bits generation, for all 16 bit,32 bit, 64 bit, 128 bit (and so on) data with check bit respectively 4, 5 6 (and so on). In section 4, we present our algorithms for error correction and detection, validating the mathematical model used in Section 3 for mitigating soft error. In section 5, we will analysis our results and showed relative efficiency of the other ECC schemes. Section 6 concludes the paper.

II. BACK GROUND OF THE CODING SCHEME

The technique for error detection and correction algorithms presented in this paper had been implemented by Reed-Muller matrix, which is used mostly to convert between Fixed Polarity Reed-Muller form and Boolean functions [8, 9]. The process is mainly divided into two parts. The algorithm which is relatively simple involves transmitting data with multiple check bits and decoding the associated check bits when receiving or when reading data from random access memory (RAM) to detect the errors [6]. By combing the exact data bits in the original message using XOR operator each check bits are generated. So at first we have generated the Reed-Muller matrix and then from the matrix we have to find out the check bit. The minimum number of check bits required to detect a single bit in the message is given by the following equation:

$$D+C+1 \leq 2r \qquad (1)$$

Where D is the number of data bits and C is the number of check bits. Reed-Muller (RM) Codes are a family of linear error-correcting codes used in communications and are one of the oldest error correcting codes. On the other hand, error correcting codes play a significant function in computational complexity theory and are very functional in sending information over extensive distances in which errors might take place in the message [7]. This section describes a theory for encoding the message “the transmitted data” by using Reed-Muller basic matrix [10, 11]. In order to generate the Reed-Muller matrix for

all 16 bit, 32 bit, 64 and 128 bit data, the following Reed-Muller matrix is used:

$$RM = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad (2)$$

To generate check bits we need to find out the corresponding 1 in the matrix and then XOR with them which will reflects the nature of the error.

III. Algorithm for Matrix and Check Bit Generation

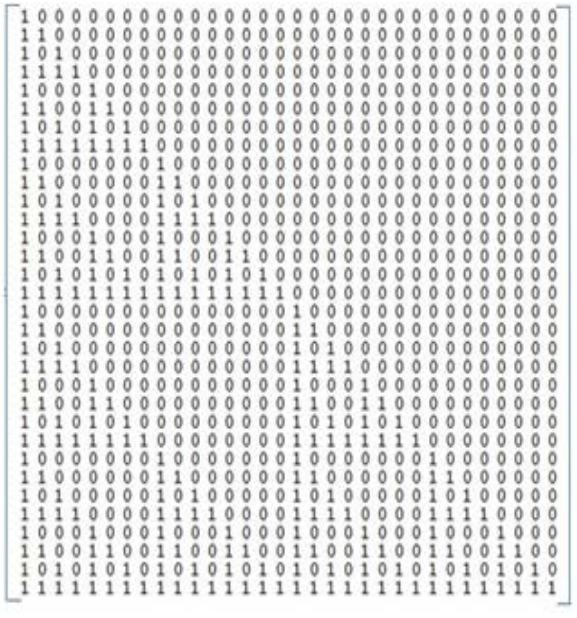
To explain the concepts involved we developed a code that can detect and correct single bit error in 16, 32, 64, 128 bit words. We used equation (2) as a basic matrix for 16, 32, 64, 128 bit data. We used the equation for determining the data bit,

$$2^C = D \quad (3)$$

Where C is the positive integer number which means how many times we need use the Kronecker operation [10, 11] over Reed-Muller matrix or how many check bits is required to detect and correct error and D is data bit. Here as an example we will explain the 32 bit Reed-Muller matrix. To generate 32 bit Reed-Muller matrix we need Kronecker operator "*" for five times. Here, in this explanation C = 5 which means the data bit D = 32 as 2^5 = 32 for 5 times Kronecker operation over Reed-Muller matrix.

$$RM(5) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad (4)$$

RM(5) =



The Algorithm we used for Reed-Muller matrix generation is given below:

Procedure: **Reed_Muller (S, Q, R)**

1. [Initialization] Set COUNT := 0 and M :=0
2. Repeat Steps 3 to 9 for I = 0,1,….S-1
3. If COUNT%R= =0 and COUNT!=0
   Then: Set M: = M+1
4. If I % 2= = 0 then: Set P :=0
   Else Set P: =1
5. Set L:=0
6. Repeat Steps 7 to 9 for J = 0,1,….Q-1
7. Set K:=0
8. Repeat while K < 2:
   a) Set X3[I][J+K] := X1[M][L] * X2[P][K]
   b) Set COUNT := COUNT + 1
   [End Loop of Step 8]
9. Set L := L + 1
10. Return

Algorithm (Reed_Muller):

X1, X2 and X3 are two dimensional global declared matrixes and initially X1 is a basic 2x2 matrix. Finally X3 contains n x n output matrix.

1. [Initialization] Set S :=4, Q :=4 and R : =8
2. Repeat Steps 3 and 4 for I =1, 2,…..n-1
3. Call **Reed_Muller (S, Q, R)**
4. Set S := S * 2, Q := Q * 2 and R : = R * 2
5. Exit

To generate the check bits from equation (3), each row in equation (3) is operated on the data bits using the XOR and the AND operations to produce the necessary check bits that are needed in the system to detect the error in the received message or the written message. The final results are given below, where "+" are XOR operation.

$C_0=D_0$
$C_1=D_0+D_1$
$C_2=D_0+D_2$
$C_3=D_0+D_1+D_2+D_3$
$C_4=D_0+D_4$
$C_5=D_0+D_1+D_4+D_5$
$C_6=D_0+D_2+D_4+D_6$
$C_7=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7$
$C_8=D_0+D_8$
$C_9=D_0+D_1+D_8+D_9$
$C_{10}=D_0+D_2+D_8+D_{10}$
$C_{11}=D_0+D_1+D_2+D_3+D_8+D_9+D_{10}+D_{11}$
$C_{12}=D_0+D_4+D8+D_{12}$
$C_{13}=D_0+D_1+D_4+D_5+D_8+D_9+D_{12}+D_{13}$
$C_{14}=D_0+D_2+D_4+D_6+D_8+D_{10}+D_{12}+D_{14}$
$C_{15}=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7+D_8+D+D_{10}+D_{11}+D_{12}+D_{13}+D_{14}+D_{15}$
$C_{16}=D_0+D_{16}$
$C_{17}=D_0+D_{16}$
$C_{18}=D_0+D_2+D_{16}+D_{18}$
$C_{19}=D_0+D_1+D_2+D_3+D_{16}+D_{17}+D_{18}+D_{19}$
$C_{20}=D_0+D_4+D_{16}+D_{20}$
$C_{21}=D_0+D_1+D_4+D_5+D_{16}+D_{17}+D_{20}+D_{21}$

$C_{22}=D_0+ D_2 + D_4 + D_6 + D_{16} + D_{18} + D_{20} + D_{22}$
$C_{23}=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7+D_{16}+D_{17}+D_{18}+D_{19}+D_{20}+D_{21}$
$+D_{22}+D_{23}$
$C_{24}=D_0+D_8 +D_{16} +D_{24}$
$C_{25} =D_0 + D_1+D_8+D_9 + D_{16} + D_{17 +}D_{24} +D_{25}$
$C_{26}=D_0 + D_2 + D_8+D_{10} + D_{16}+ D_{18}+D_{24} +D_{26}$
$C_{27}=D_0+D_1+D_2+D_3+D_8+D_9+D_{10}+D_{11}+D_{16}+D_{17}+D_{18}D_{19}+D_{24}$
$+D_{25}+D_{26}+D_{27}$
$C_{28}=D_0 + D_4 + D_{8 +} D_{12} + D_{16} + D_{20}+ D_{24} + D_{28}$
$C_{29}=D_0+D_1+D_4+D_5+D_8+D_9+D_{12}+D_{13}+D_{16}+D_{17}+D_{20}+D_{21}+D_{24}$
$+D_{25}+D_{28}+D_{29}$
$C_{30}=D_0+D_2+D_4+D_6+D_8+D_{10}+D_{12}+D_{14}+D_{16}+D_{18}+D_{20}+D_{22}+D_{24}+$
$D_{26}+D_{28}+D_{30}$
$C_{31}=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7+D_8+D_9+D_{10}+D_{11}+D_{12}+D_{13}+$

$D_{14}+D_{15}+D_{16}+D_{17}+D_{18}+D_{19}+D_{20}+D_{21}+D_{22}+D_{23}+D_{24}+D_{25}+D_{26}$
$+$
$D_{27}+D_{28} +D_{29} +D_{30}+D_{31}$

Now we need to calculate the check bits from the C0 to C31. As in this example we are working with 32 bit data, from equation (3), we find there are five check bits. This five check bits are calculated from the following equation

$$Q = 2^{C-1} \quad (5)$$

Where Q means how many 1's are in every check bit as the sum of 1 one column in the matrix and C = No. of Check bit, C. Those we have Q number of 1 in their column they are selected as check bit. So therefore, here C = 5 and Q = 16 which means from C_0 to C_{31} who has 16 one, they should be selected as check bit. From the equation (3) and (5) we can derive the five check bit set which are as follows.

$C_{15}=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7+D_8+D+$
$D_{10}+D_{11} +D_{12}+D_{13}+D_{14}+D_{15}$
$C_{23}=D_0+D_1+D_2+D_3+D_4+D_5+D_6+D_7 +$
$D_{16}+D_{17}+D_{18}+D_{19}+D_{20}+D_{21}+D_{22}+D_{23}$
$C_{27}=D_0+D_1+D_2+D_3+D_8+D_9+D_{10}+D_{11}+D_{16}+$
$D_{17 +} D_{18}D_{19}+D_{24} +D_{25}+D_{26}+D_{27}$
$C_{29}=D_0+D_1+D_4+D_5+D_8+D_9+D_{12}+D_{13}+D_{16}+$
$D_{17} +D_{20}+D_{21}+D_{24} +D_{25}+D_{28}+D_{29}$
$C_{30}=D_0+D_2+D_4+D_6+D_8+D_{10}+D_{12}+D_{14}+D_{16}+$
$D_{18}+D_{20}+D_{22}+D_{24}+D_{26}+D_{28}+D_{30}$

Now, we will denote $C_{15,}$ $C_{23,}$ $C_{27,}$ $_{C29}$ and C_{30} respectively to C_0, C_1, C_2, C_3 and C_4.

From equation (3) for 64 data bit, the no of check bit is 6 as $64 = 2^6$ and from the equation (5) the check bit set are as follows as each of the following contains 32 one in each column, C_{31}, C_{47}, C_{55}, C_{59}, C_{61} and $C_{62.}$

Table 1: The Corresponding check bits for 32 data bit, D_0 to D_{31}

Data bits	Generated check bit				
	C_0	C_1	C_2	C_3	C_4
D_0	×	×	×	×	×
D_1	×	×	×	×	
D_2	×	×	×		×
D_3	×	×	×		
D_4	×	×		×	×
D_5	×	×		×	
D_6	×	×			×
D_7	×	×			
D_8	×		×	×	×
D_9	×		×	×	
D_{10}	×		×		×
D_{11}	×		×		
D_{12}	×			×	×
D_{13}	×			×	
D_{14}	×				×
D_{15}	×				
D_{16}		×	×	×	×
D_{17}		×	×	×	
D_{18}		×	×		×
D_{19}		×	×		
D_{20}		×		×	×
D_{21}		×		×	
D_{22}		×			×
D_{23}		×			
D_{24}			×	×	×
D_{25}			×	×	
D_{26}			×		×
D_{27}			×		
D_{28}				×	×
D_{29}				×	
D_{30}					×
D_{31}					

IV. Algorithm for Error correction and detection:

Here we present the two algorithms for research of this paper for Error detection and correction for single data bit error.

Algorithm EDC: Error Detection for data bit error () {

1. Calculation of the Syndrome Code,
   Syndrome Code = Received Check bit XOR Receiver generated Received Check bit.

2. IF Syndrome Code contains only zero, then there is no Error detected in Received Check bit. Do nothing or Print NO Error.
3. IF Syndrome Code contains multiple one but not all one, Then Error detected in one of the Received Data bit. Print Error in Data bit.

4. IF Syndrome Code contains all one, Then Error detected in the first Received Data bit. Print Error D_0 Data bit.
   }

Algorithm ECC: Error Correction for data bit error () {

1. IF Error detected in the first Received Data bit, then inverse the first Received Data bit.

2. IF Error detected in one of the Received Data bit, then the sum or weight of numerical value of the data bit is calculated by identified the zero-bits in the syndrome starting from the second bit left to right. These numerical values are zero and power of 2 as 2^N where N = 0, 1, 2, 3...to C, C = No. of check bit.

3. After identified the position of error in the Received Data bit from 2, the data bit position is inversed.
}

V. Implementation and Result Analysis:

In paper [6], they proposed the error correction and detection code for 16 data bit with 6 check bits. Here, in our research, we showed the method which will work on 16, 32, 64 and 128 data bits and the check bits required respectively 4, 5, 6 and 7 which will reduce the memory cost magnificently.

Our unique contribution of the paper is the equation (3) and (4) which we proposed here to identified the check bits.

Table 2: Identifying the check bits

2^C	Data Bits, D	Check bit, C	C contains 2^{C-1}'s 1(one) in each Colum
2^4	16	4	8
2^5	32	5	16
2^6	64	6	32
2^7	128	7	64

In our Error Correction Code Algorithms which we proposed here is very faster than [6], as we introduced power of 2 to identify the error position where one of the Received Data bits are error.

From the algorithm we have simulated the error in data bit in our own simulator and find the result for 32 bit data, which is given in Table 3. Table 3 summarizes the process if one of the data bit become corrupted. Here the data massage = 01110011011001010101001101101101 and check bit = 01111. We get the Syndrome by the following equation:

Syndrome = Received check bits + Recalculated/Receiver Generated Check bits (6)

Where "+" means XOR operation and we got recalculated check bits from equation (5).

Table 3: Syndrome with recalculated check bits and error bit position of the error data

Data	Received check bits $C_4\ C_3\ C_2\ C_1\ C_0$	Recalculated check bits $C_4\ C_3\ C_2\ C_1\ C_0$	Syndrome	Error bit position of data
D_0	01111	10000	11111	0
D_1	01111	00000	01111	1
D_2	01111	11000	10111	2
D_3	01111	01000	00111	1+2=3
D_4	01111	10100	11011	4
D_5	01111	00100	01011	1+4=5
D_{16}	01111	10001	11110	16
D_{17}	01111	00001	01110	1+16=17
D_{18}	01111	11001	10110	2+16=18
D_{19}	01111	01001	00110	1+2+16=19
D_{20}	01111	10101	11010	4+16=20
D_{27}	01111	01011	00100	1+2+8+16=27
D_{28}	01111	10111	11000	4+8+16=28
D_{29}	01111	00111	01000	1+4+8+16=29
D_{30}	01111	11111	10000	2+4+8+16=30
D_{31}	01111	01111	00000	1+2+4+8+16=31

To find the error in one of the check bits, the following Table 4 is used:

Table 4: Syndrome for error check bit

Check bit	Received check bits $C_4\ C_3\ C_2\ C_1\ C_0$	Recalculated check bits $C_4\ C_3\ C_2\ C_1\ C_0$	Syndrome	Error bit position of check bit
C_0	01110	01111	00001	0
C_1	01101	01111	00010	1
C_2	01011	01111	00100	2
C_3	00111	01111	01000	3
C_4	11111	01111	10000	4

VI. CONCLUSION

In this paper we simply introduced a simple algorithm, which can be used to detect, and correct the errors in the transmitted data and check bit separately based on Reed-Muller matrix. The algorithm was tested on single bit error in our designed simulator and has found correct results for 16, 32, 64 and 128 data bits. In Future the algorithm can be extended to find more than single bit error.

REFERENCES

[1] Pramod S P, Rajagopal A, Akshay S Kotain,'FPGA Implementation of Single Bit Error Correction using CRC', *International Journal of Computer Applications (0975 – 8887) Volume 52– No.10, August 2012*

[2] [2] Nidhi Syal, Vaneet Chahal, 'Comparison of "reed_muller and hamming codes" for error detection and correction', *ISP Journal of Electronics Engineering, Vol1, Issue 1, October'2011, (5-9)*

[3] Md. Sharif Uddin, Cheol Hong Kim, Jong-MyonKim, 'Accelerating Soft-Decision Reed-Muller Decoding Using a Graphics Processing Unit', Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology Vol.4, No.2, December (2014), pp. 369-378

[4] Varsha P. Patil, Prof. D. G. Chougule, Radhika.R.Naik, 'Viterbi Algorithm for error detection and correction', IOSR Journal of Electronicsl and Communication Engineering (IOSR-JECE) ISSN: 2278-2834-, ISBN: 2278-8735, PP: 60-65

[5] Notes on 'Error Detection and Correction ', *Version 2 CSE IIT, Kharagpur*

[6] Khalid Faraj, 'Design Error Detection and Correction System based on Reed_Muller Matrix for Memory Protection', *International Journal of Computer Applications (0975 – 8887) Volume 34– No.8, November 2011*

[7] Othman O. Khalifa, Aisha-Huassan Abdullah, N. Suriyana, Saidah Zawanah and Shihab A. Hameed 'Reed-Muller Codec Simulation Performance', *Journal of Computer Science 4 (10): 792-798, 2008 ISSN 1549-3636*

[8] Reed, I. S., 'Class of multiple error correcting codes and their decoding scheme', Institute of Radio Engineers Transaction on Information Theory, PGIT-4: pp. 38- 49, 1954.

[9] Muller, D. E., 'Application of Boolean algebra to switching circuit design and to error detection', *Institute of Radio Engineers Transaction on Electronic Computers, EC-3: pp. 6-12, September 1954.*

[10] Faraj, K., Almaini, A.E.A.,'Minimization of Dual Reed-Muller Forms using Dual Property' *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Issue 1, Vol. 6, pp.9-15, January 2007.*

[11] Faraj, K., Almaini, A.E.A., 'Optimal Expression for Fixed Polarity Dual Reed-Muller Forms', *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Issue 3, Vol. 6, pp.9-15, March 2007.*

AUTHORS

First Author – Monika Gope, M.Sc (CSE ongoing), Khulna University of Engineering & Technology, gopenath14@yahoo.com.

Second Author – Md. Ariful Islam Khandaker, M.Sc(CSE ongoing), Khulna University of Engineering & Technology,aikhandaker@yahoo.com.

Third Author – Mehnuma Tabassum Omar, M.Sc(CSE ongoing), Khulna University of Engineering & Technology, misty2409@gmail.com.