Design Error Detection and Correction System based on Reed_Muller Matrix for Memory Protection

Khalid Faraj
Birzeit University
Department of Computer Systems Engineering
Birzeit, Palestine

ABSRACT
This paper describes a new method to detect and correct a single bit in the data message. This method has been developed based on Reed Muller matrix. The key point for the implementation of error-free is the encoding of the information to be transmitted in such a way that some extent of redundancy is included in the encoded data, and a method for efficient decoding at the receiver is available. These two requirements have been achieved in the new method in an efficient and simple way. The new method is implemented using XILINX, and has demonstrated using some examples. The design detects and corrects all single bit errors in a 16 bit data, and 6 check bits.

General Terms
Error detection, Reed Muller, Coding.

Keywords
Communication, coding, encoding messages, corrects the message.

1. INTRODUCTION
Error detection and correction is found in many reliable and performance applications, such as data storage systems, random access memory (RAM), SRAM, and cache memory. Electrical or magnetic interference inside a computer system can cause a single bit of RAM to spontaneously flip to the opposite state. This is mainly due to alpha particles emitted by contaminants in chip packaging material, or of background radiation, chiefly neutrons from cosmic rays [1-3], which may change the contents of one or more memory cells or interfere with the circuitry used to read/write them, therefore, there is a need for coding theory in order to have a reliable communication over an unreliable channel. All solutions to this problem, in some form or another, depend on the basic idea of encoding messages with some redundancy, allowing the receiver to detect and correct whatever errors may arise during transmission through the channel. The main goal is to minimize the amount of redundancy while maximizing the quantity of errors that can be corrected. Networks and other communication systems must be able to transfer data from the source to the receiver with complete accuracy. A system that cannot guarantee that the data received by one device are identical to the data transmitted by another device is essentially useless.

The goal of this research is to design coding schemes which are capable of detecting and correcting such errors. The setting is usually modeled as follows: a transmitter starts with some message, which is represented as a string of symbols over some alphabet. The transmitter encodes the message into a longer string over the same alphabet, and transmits the block of data over a channel. The channel introduces errors (or noise) by changing some of the symbols of the transmitted block, and then delivers the corrupted block to the receiver. Finally, the receiver attempts to decode the block, hopefully to the intended message. Whenever the transmitter wants to transmit a new message, the process is repeated. Two factors are of special interest in this setting. The first is the information rate, which is the ratio of the message length to the encoded block length. This is a measure of how much actual message data is carried by each transmitted symbol. The second is the error rate, which is the ratio of the number of errors to the block length. This is a measure of how noisy the channel is, i.e. how much data it corrupts. Of course, we desire coding schemes that tolerate high error rates while simultaneously having large information rates. In practice, smaller alphabets are desirable too, as most digital communication devices are, at their lowest levels, capable of interpreting only binary digits (bits) [4, 5].

Shannon demonstrated how information can be encoded to withstand such noise with probability arbitrarily close to 1 [6]. Two years later on error-correcting codes [7], Hamming proposed an adversarial channel that perturbs symbols in a worst-case fashion. This model of a channel is much more "pessimistic" than Shannon's, as it encompasses any arbitrarily-complex source of noise. As such, it gives much stronger guarantees on the robustness of the resulting coding scheme.

2. CODING ALGORITHM
Error detection and correction functions described in this paper are made possible by Reed-Muller matrix, which is used basically to transform between Fixed Polarity Reed-Muller form and Boolean functions [8, 9]. The algorithm is relatively simple yet powerful code. It 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. The check bits are generated in parallel form, by combing certain data bits in the original message using XOR operator for each check bit. The combination of these check bits errors reflects the nature of the error. The minimum number of check bits required to detect a single bit in the message is given by the following equation:

$$D + c + 1 \leq 2^c \qquad (1)$$

Where D is the number of data bits and C is the number of check bits.

In the communication system, we are concerned mostly with the encoder in the transmitter part. The encoder adds some extra bits to the original message, which are used in the receiver part in order to detect the error which has happened either in the message or in the check bits themselves. Hence, the main function of the decoder is to detect the errors, corrects errors, or the combination of both.

To clarify the concepts involved we will develop a code that can detect and correct single bit error in 16 bit words.

This section describes a new 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 16 bits data, the following Reed-Muller matrix is used:

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

Equation (2) is the basic matrix for generating the check bits for 16 bit data. To generate Reed-Muller matrix for 16 bits, equation (2) has to operated four times using Kronecker operator '*' [10,11],which yields the following matrix:

$$RM(4) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad (3)$$

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

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 in equation (4).

$C_0 = D_0$
$C_1 = D_0 \oplus D_1$
$C_2 = D_0 \oplus D_2$
$C_3 = D_0 \oplus D_1 \oplus D_2 \oplus D_3$ (4)
$C_4 = D_0 \oplus D_4$
$C_5 = D_0 \oplus D_1 \oplus D_4 \oplus D_5$
$C_6 = D_0 \oplus D_2 \oplus D_4 \oplus D_6$
$C_7 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7$
$C_8 = D_0 \oplus D_8$
$C_9 = D_0 \oplus D_1 \oplus D_8 \oplus D_9$
$C_{10} = D_0 \oplus D_2 \oplus D_8 \oplus D_{10}$
$C_{11} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$
$C_{12} = D_0 \oplus D_4 \oplus D_8 \oplus D_{10} \oplus D_{12}$
$C_{13} = D_0 \oplus D_1 \oplus D_4 \oplus D_5 \oplus D_8 \oplus D_9 \oplus D_{12} \oplus D_{13}$
$C_{14} = D_0 \oplus D_2 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{12} \oplus D_{14}$
$C_{15} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15}$
$C_{31} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15} \oplus \ldots \oplus D_{31}$

Where '⊕' is the Exclusive-OR function.

Note: C_{31} does no need to include D_{16} to D_{31}, because this part of data does not exist hence, C_{31} is identical to C_{15} and there is no need to calculate it.
The check bits that are needed to correct the error in the message or in the memory from equation (4) are the following set:

$C_7 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7$
$C_{11} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$ (5)
$C_{13} = D_0 \oplus D_1 \oplus D_4 \oplus D_5 \oplus D_8 \oplus D_9 \oplus D_{12} \oplus D_{13}$
$C_{14} = D_0 \oplus D_2 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{12} \oplus D_{14}$
$C_{15} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15}$
$C_{31} = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15}$

For convenient we shall rename $C_{31}C_{15}C_{14}C_{13}C_{11}$ C_7 to C_5 $C_4C_3C_2C_1C_0$.

Table 1 shows the participating check bits

Participating Data bits	Generated check bits					
	C_0	C_1	C_2	C_3	C_4	C_5
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}					×	×

The following example demonstrates how to calculate the check bits:

Example one:
Let the number of the transmitted data (*k*) is sixteen bits ($D_{15}D_{14}D_{13}D_{12}D,\ldots\ldots D_0$) = (0111001101001101), The encoder will generate the required code, which consists of the check bits (***c***). Therefore, the transmitter sends the information bits (*k*), and the check bits vector.
The generated check bits are constructed as follows:

Using equation (5):

$C_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7$
$C_0 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$
$C_1 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11}$
$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$
$C_2 = D_0 \oplus D_1 \oplus D_4 \oplus D_5 \oplus D_8 \oplus D_9 \oplus D_{12} \oplus D_{13}$
$C_2 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$
$C_3 = D_0 \oplus D_2 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{12} \oplus D_{14}$
$C_3 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$
$C_4 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15}$
$C_4 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$
$C_5 = C_4$

Therefore, the transmitted check bit vector ($C_5C_4C_3C_2C_1C_0$) is (110110).

3. ERROR CORRECTION USING A REED-MULLER MATRIX

This section describes in detail the correction process. This process can be used either in the receiver side or in correcting errors in semiconductor memory chips.
The received message, which is composed of data bits and check bits, where the error bits are inserted at the end of message. The only crucial point in the selection of their locations is that both the sender and receiver are aware of where they actually are.
Once the message with the error bits are received from the sender or from the memory chips, the receiver will generate a new set of check bits. The generated bits are combined again with the received check bits in order to determine the error and the location of the error. The result of this combination is called the syndrome.

For convenience, the algorithm generates a 6-bit syndrome for a 16-bit data word with the following characteristics:

1. If the syndrome contains one and only one bit set to one, then there is an error in one of the check bits.

2. If the syndrome contains all zeroes, then there is no error, and the transmitted data is accurate.

3. If the syndrome contains more than one bit set to one, then there is an error in one of the data bits. The numerical value of the data bit is calculated by identified the zero-bits in the syndrome. The weight of each zero-bit in the syndrome starting from the second bit from left to right in the decimal value is (0,1,2,4,8) as shown in table (2). Except the first case in the syndrome, where all of its bits are ones then the error is in the first data bit.

It is best to illustrate the detection process and view the steps in generating the syndromes by using an example.

Example 2:
Determine the new check bits for the message which was used in example one ($D_{15}D_{14}D_{13}D_{12}D,\ldots\ldots D_0$ $C_5C_4C_3C_2C_1C_0$) (0111001101001101, 110110) .
In the receiver side, we will recalculate the chick bits again using equation (5) in the same way as in the transmitter.

Then the new chick bits are compared with the received chick bits as the following:

1. Suppose now that data D_0 sustains an error and is changed its value from 1 to 0.

Then the new chick bits according to this data are recalculated as before to yield the following:

$C_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7$
$C_0 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 1$
$C_1 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D11$
$C_1 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$
$C_2 = D_0 \oplus D_1 \oplus D_4 \oplus D_5 \oplus D_8 \oplus D_9 \oplus D_{12} \oplus D_{13}$
$C_2 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 0$
$C_3 = D_0 \oplus D_2 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{12} \oplus D_{14}$
$C_3 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$
$C_4 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15}$
$C_4 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 0$
$C_5 = C_4$

Hence, the regenerated code is (001001)

To determine whether the error has occurred in one of data bits or in the check bits or none of that, the new check bits are compared with the old check bits using XOR operation, in order to generate the syndrome word as the following:

Transmitted error code: $\oplus C_5C_4C_3C_2C_1C_0$
Regenerated error code: $C_5C_4C_3C_2C_1C_0$
Syndrome code:
This gives the following result:

$\oplus 110110$
$\underline{001001}$
111111

The result is 111111, indicating that data (D_0) is wrong; it should be one and not zero.

2. Suppose now that data D_1 sustains an error and is changed its value from 0 to 1.

The new check bits are recalculated according to this data (0111001101001110, 011011) using equation (5) to yield the following syndrome:

Transmitted error code: $\oplus C_5C_4C_3C_2C_1C_0$
Regenerated error code: $C_5C_4C_3C_2C_1C_0$

Then the new are:

$C_5C_4C_3C_2C_1C_0 = 00001$

This produces the following syndrome:

$\oplus 110110$
$\underline{000001}$
110111

The syndrome word is $(p_0p_1p_2p_3\,p_4) = 110111$,

Hence, the weight of each bit from left to right in the decimal value is (0,1,2,4,8). Therefore, the value of the corrupted bit reading just the zero bits in the syndrome is one.

This process is carried out by using the same procedure to find the reset of the syndromes for each of the corrupted bit as follow:

For D_2 the new syndrome is:

Transmitted error code: $\oplus$ 110110
Regenerated error code: $\underline{001101}$
Syndrome code: 111011

Therefore, the value of the corrupted bit is in D_2 according to the location of zeros in the syndrome.

For D_3 the new syndrome is:

Transmitted error code: $\oplus$110110
Regenerated error code: $\underline{000101}$
Syndrome code: 110011

Therefore, the value of the corrupted bit is in D_3 according to the location of zeros in the syndrome is one plus two gives three.

3. For D_4 the new syndrome is:

Transmitted error code: $\oplus$110110
Regenerated error code: $\underline{001011}$
Syndrome code: 111101

Therefore, the value of the corrupted bit is in D_4 according to the location of zeros in the syndrome is four.

Table (2) summarize the process if one of the transmitted data has been corrupted, gives the corresponding syndrome, and the data bit number or location.

Table 2 Syndromes for corrupted data bits

Data	Check bits received	Recalculated Check bits	Syndrome	Corrupted bit
D_0	110110	001001	111111	0
D_1	110110	000001	110111	1
D_2	110110	001101	111011	2
D_3	110110	000101	110011	1+2=3
D_4	110110	001011	111101	4
D_5	110110	000011	110101	1+4=5
D_6	110110	001111	111001	2+4=6
D_7	110110	000111	110001	1+2+4=7
D_8	110110	001000	111110	8
D_9	110110	000000	110110	1+8=9
D_{10}	110110	001100	111010	2+8=10
D_{11}	110110	000100	110010	1+2+8=11
D_{12}	110110	001010	111100	4+8=12
D_{13}	110110	000010	110100	1+4+8=13
D_{14}	110110	001110	111000	2+4+8=14
D_{15}	110110	000110	110000	1+2+4+8=15

As shown in Table (2) the algorithm is able to detect and locate its location in order to fix it in the message or in the memory chips.

To find the error in one of the check bits, the following algorithm is introduced:

If the error has occurred in C_0, the syndrome is calculated as follows using the same data as in example (1):

Transmitted error code: $\oplus\ C_5C_4C_3C_2C_1C_0$
Regenerated error code: $C_5C_4C_3C_2C_1C_0$

Transmitted error code: $\oplus$ 110111
Regenerated error code: 110110
Syndrome code: 000001

The result reflects an error occurred in the first check point, and only one of the syndrome bits is set to one, and the rest of the bits are zeros.

Suppose now that C_1 has an error, then the syndrome is:

Transmitted error code: $\oplus$ 110100
Regenerated error code: 110110
Syndrome code: 000010

Therefore, the second bit is set to one, which reflects the second check bit is wrong.

The following table (3) summarizes the process if one of the check bits has been corrupted. It gives the corresponding syndrome, and the check bit number.

Table 3: Syndromes for corrupted check bits

Error in check bit	Check bits received	Recalculated Check bits	syndrome	Corrupted bit
C_0	110111	110110	000001	C_0
C_1	110100	110110	000010	C_1
C_2	110010	110110	000100	C_2
C_3	111110	110110	001000	C_3
C_4	100110	110110	010000	C_4
C_5	010110	110110	100000	C_5

We summarize the final characteristics for the five bits syndromes which were generated by XORing the received check bits with the recalculated check bits as follows:

4. CIRCUIT DESIGN

In this section we describe the main parts to implement the algorithms in this paper. The integrated circuit can be used on a Bus to detect a single bit error in memory data and correct that bit. The circuit can be further improve to detect a double bits error by increasing the check bits by one, but it will not be able to correct the error. Figure 1 shows the block diagram of Error Detection and Correction System. The proposed device is 16_bit parallel Error Detection and Correction Circuit (EDCC). The EDCC uses a Reed_Muller matrix to generate a 6 bit check

word from a 16 bit data word. The check bits are stored along with the data word during the memory write phase.

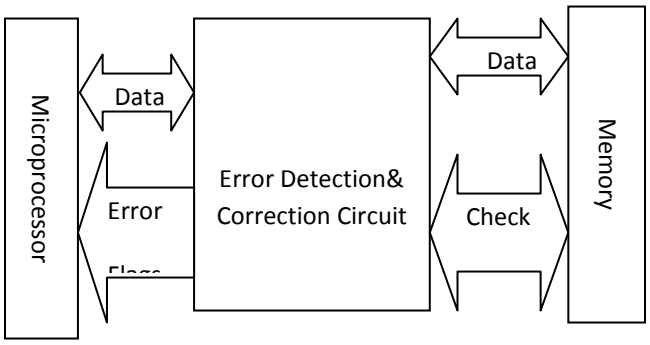

Figure 1: Error Detection & Correction System

5. MEMORY READ CYCLE

During a memory read cycle, the 16-bit data along with the 6-bit check word are retrieved. To determine whether the data word from the memory is acceptable to use as presented on the bus, the error flags must be tested first. Table 4 shows the error function according to the two flags (EC and ED), where the EC is the error flag for check bits while the ED is the error data flag.

Table 4. Error Function

Error Flags EC ED	Type of Error
0 0	No error in data
0 1	Error in data
1 0	Error in Check bit
1 1	Not applicable

The first case in Table 4 represents no error condition. The second case represents an error case in one of data bits; the error bit must be located and corrected in the next parts of the circuit. The third case indicates that one of the check bits is corrupted and no further action is taken, while the last case is not applicable.

During the read cycle, the data and the check bits stored in the memory are input through the input ports. New check bits internally are generated from the data bits in the check bit generator section. The new check bits are compared with the stored check bits by an Exclusive OR operation in the Syndrome generator to produce the syndrome word. Figure (2) shows the switching waveforms during the read and correct mode.

.

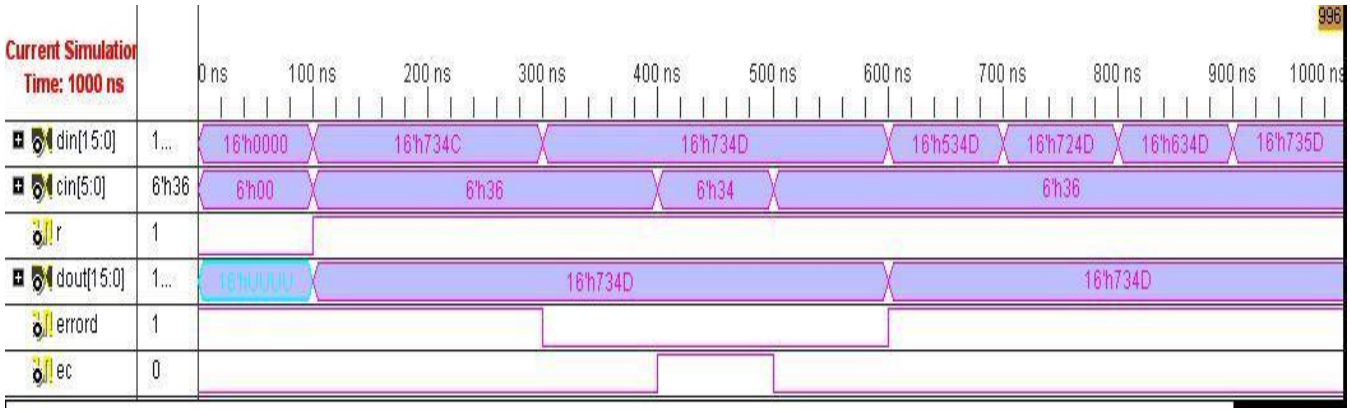

Fig 2: Read, and Correct Mode

6. WRITE CYCLE

The proposed Error Correcting Code (ECC), shown in Fig. 1. The operation of the ECC is quite simple. During the write cycle, data is written along with check bits. The check bits are generated from the data bits in the Check Bit Generator section as seen from Table 2. The table is constructed according to equation (4) the effect of each erroneous bit is unique. The unique combination of parity check bits is produced in each individual case; therefore, the erroneous bit can be easily detected.

7. CONCLUSION

In this paper we introduced a simple algorithm, which can be used to detect, and correct the errors in the transmitted message based on Reed-Muller matrix. The algorithm was tested on some examples, and has given correct results. The algorithm can be extended to *n* bits messages, and can be implemented on integrated circuits based on XOR gates.

8. ACKNOWLEDGMENTS

I would like to thank the Royal Society of Edinburgh for their support.

9. REFERENCES

[1] K. Osada, Y. Saitoh, E. Ibe, K. Ishibashi. 16.7fA/cell Tunnel_Leakage Suppressed 16Mb SRAM for Handling Cosmic ray Induced Multi_Errors, ISSCC Dig. Tech. Papers, pp. 302_303, Feb. 2003

[2] T. Suzuki et al, 0.3 to 1.5V Embedded SRAM with Device Fluctation Tolerant Access control and Cosmic Ray Immune Hidden ECC Scheme. ISSCC Dig. Tech. Papers, pp. 484_612, Feb. 2005

[3] T. Suzuki et al, "0.3 to 1.5V Embedded SRAM with Device-Fluctuation-Tolerant Access-Control and Cosmic-Ray-Immune Hidden ECC Scheme", ISSCC Dig. Tech. Papers, pp. 484-612, Feb. 2005

[4] B. P. Lathi., 1983, Modern Digital and Analog Communication Systems. CBS College Publishing.

[5] Simon Haykin, 1994, Communication Systems. John Wiley & Sons, INC.

[6] Claude E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27:379–423, July 1948.

[7] Richard W. Hamming. Error detecting and error correcting codes. Bell System Technical Journal, 29:147–160, 1950.

[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.