Improving Performance Parameters of Error Detection and Correction in HDLC Protocol by using Hamming Method

Varinder Singh
Electronics and Communication Engineering, Amritsar College of Engineering and Technology, Amritsar, Punjab, India

Narinder Sharma
H.O.D. Department of Electrical and Electronics Engineering, Amritsar College of Engineering and Technology, Amritsar, Punjab, India

ABSTRACT
This senate proposes an optimize version of the Hamming codes for Error detection and correction (EDAC) used in the HDLC protocol as HDLC being the most enduring and fundamental standard in communication. When data is either stored in memory or transmitted through a communication channel it is not errorless. With the exposure to electromagnetic radiation the semiconductor memory which are used in various applications get damaged .Because of electromagnetic interference the contents of the RAM memory cells get affected which leads to the bit flip in the magnetic storage devices like floppy disk, magnetic tape and hard disk devices etc. A proposed method has been developed to overcome the existing problems by using Xilinx ISE 13.2 simulator tool through which number of bit errors detection and correction can be increased in 8x8 matrix .It will result into enhancement of code rate and reduction of bit overhead.

Keywords
Error detection and correction (EDAC), High level data link control (HDLC) Hamming method.

1. INTRODUCTION
During transmission of data or in read, write operation from the memory it is necessary to ensure that data has not been corrupted either by noise or by any hardware failure various EDAC techniques are employed. This is because different codes are being used in different applications. Due to classical development of the memory units different types of errors generated and the kind of overhead associated with different error detection techniques the main requirement of error detection and correction method is as follow :

(a) Network must transfer the error free data between transmitter and receiver.

(b) Errors need to be detected and corrected for reliable communication. Lastly, error detection and correction must be deployed at the data link layer of the OSI model[4].

Source coding and channel coding can be used to improve the reliability and efficiency of data transmission. Source coding is employed for the reliability during transformation of data from one form to another and channel coding is used for efficiency when the data is made secure from noise by making use of error detection and correction techniques[1,3]. Although numerous EDAC methods have been developed to protect data from errors in the last few decades but still the reliability in the data transmission is a big concern in the noisy environment.

This paper represents a highly efficient and reliable method for error detection and correction. The proposed EDAC methodology is based on HDLC protocol using Hamming method. In this method the HDLC protocol protects the data part from error with the use of Hamming code and can correct more number of errors than any other EDAC method from the same amount of data bits[5].

1.1 Code Rate
The code rate is defined as ratio of amount of information bits (m) to the total number of bits that are transmitted in the codeword. It is a significant part for the evaluation of performance of any error detecting and correcting code. For a better error detection and correction method it should have more code rate.

Code rate (R) = Data bits(m)/Total number of bits (n + m).(1)

1.2 Bit Overhead
Another significant term for the comparison of two codes is the bit overhead. The bit overhead is defined as the ratio of parity bits to the data bits present in the codeword. Bit Overhead determines the percentage of redundancy in the codeword. A better error detection and correction method should have less bit overhead.

Bit Overhead (BO) = Parity bits (n) / Data bits (m) (2)

2. RELATED WORK
Golay codes are an important example of cyclic codes. The binary linear code G11 is a *(*11, 6, 5*)* code which Consists of $3^5 = 243$ distinct code words. Each of this codeword is 11 bits long and having hamming distance 4. G11 *(*11, 6, 5*)* is widely used in space applications [12]. In fact, it is an example of perfect binary code [26]. It can perform error detection and correction through using table lookup in a directory [13, 11]. This code associates 6 data bits with 5 parity bits to form a set of $3^5 = 243$ code words each of which is 11 bit long. The hamming distance is 4. Therefore, all error patterns up to three errors can be corrected. We used bit overhead and code rate for comparing different error correction and error detection methods. Bit overhead is the rate of parity bits to the number of data bits. Code rate is the number of data bits to number of bits in codeword. As a result, a method is better to have a less bit overhead and more code rate.

Bit Overhead = number of parity bits/ number of data bits
= 5/ 6 = 83.33% By using equation (1)

Code Rate = number of data bits /number of codeword

= 6/11 = 54.54% By using equation (2)

BCH codes [2] belong to class of linear cyclic block codes. For any integer $m \geq 3$, and $t < 2m - 1$, there exists a primitive BCH code characterized by following parameters.

Block length : $n = 2^m - 1$,

Number of parity check digits: $n-k \leq mt$,

Minimum Distance : $dh \geq 2t + 1$

For $m = 4$ and $t = 2$, there exists BCH *(*15, 8, 5*)* code which has triple error correction capability. In order to avoid unnecessary explanation, interested reader is referred to [2,19] for detailed information. The code rate and bit overhead for BCH *(*15, 8, 5*)* would be as follows:

Bit Overhead = 7/8 = 87.50% Using equation (1)

Code Rate = 8/15 = 53.33% Using equation (2)

Reed-Solomon [24] is a special case of BCH code. It is a [n, k, $n - k + 1$] code; in other words, it is a linear block code of length n with dimension k and minimum Hamming distance $n - k + 1$. The error-correcting ability of a Reed–Solomon code is determined by its minimum distance, or equivalently, by $n-k$, the measure of redundancy in the block. If the locations of the error symbols are not known in advance, then a Reed–Solomon code can correct up to $(n-k)/2$ erroneous symbols i.e., it can correct half as many errors as there are redundant symbols added to the block.

For $n = 31$ and $k = 26$, Reed-Solomon can correct up to 3 bit upsets in data.

The code rate and bit overhead for RS *(*31, 26, 6*)* would be as follows:

Bit Overhead = 6/26 = 23.07% Using equation (1)

Code Rate = 26/31 =83.87% Using equation (2)

Rectangular parity codes are extension of simple parity codes. By assuming the logical organization of memory bits in rectangular form and calculating a parity check bit at the end of each row and each column, we get a two dimensional parity check matrix. In the rectangular parity codes, when any data bit is changed, the corresponding column parity bit and row parity bit is changed. This scheme can easily detect single bit-upset considering any changes in the vertical and horizontal parity bits. It also can detect and in some cases correct the multiple-bit upsets in the data part using simple algorithms. It should be noted that rectangular parity codes are not able to correct or even detect some kinds of arrangement of multiple-bit upsets. However, if there is an error in the parity bits, the scheme may encounter some problems in detecting and correcting the error.

3. PROPOSED METHODOLOGY

This technique proposed an optimized version of the hamming code. The architecture presented below can be implemented with Hamming parity method. At the transmitter side, the data which is to be transmitted will be first read from the stored 8x8 RAM location and then with the Hamming parity calculation by adding parity bits to the data bits is transferred in HDLC Frame generator. After frame formation, data will be transmitted over the channel from the transmitter to the receiver. The clock will be used for required processes only and for the rest of the time it will remain ideal. At the receiver side, the HDLC frame will be received and then after deformation of HDLC frame the data with parity bit is passed to the Hamming parity calculator for parity bit calculation then check for errors if any error is detected then it is corrected . At the last, the data is written on RAM memory at desired location.

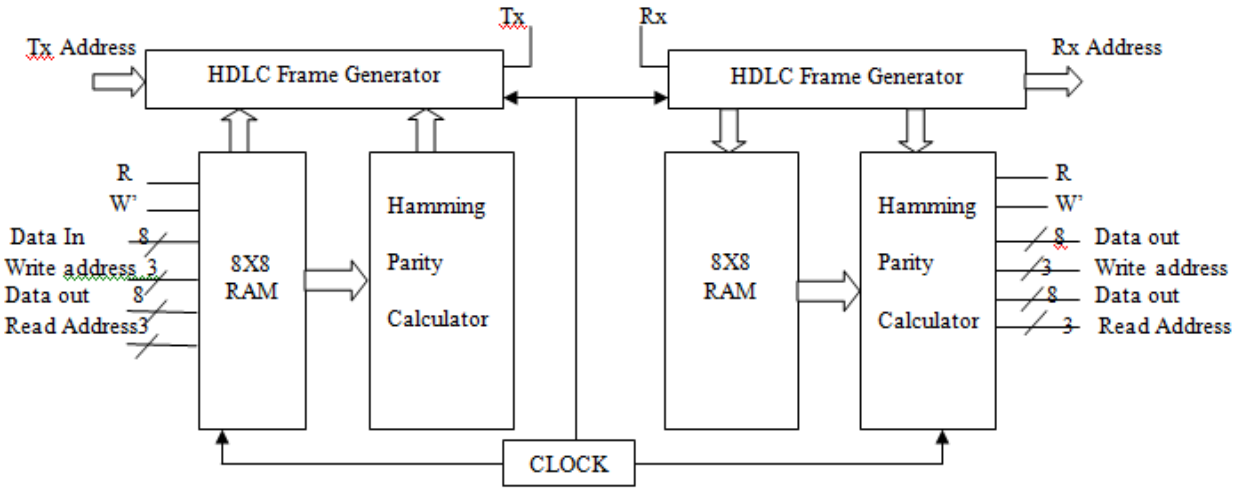

Fig. 1. Proposed Architecture

3.1 Transmitter Section

At the rising edge of the clock, the counter will check for its value. If it is equal to zero (0) then w_tx, r_tx and tx_rx_en signals are checked, either the value of signals w_tx=1,r_tx=0 and tx_rx_en=0 then perform the write data operation in the RAM at w address or the value of w_tx=0,r_tx=1 and tx_rx_en=0 then perform read data operation at RAM from r address.After this Hamming parity calculation operation is done.Parity of RAM data firstly checked and then stored in the parity storage.

To calculate parity at the transmitter side , the most common types of error correcting codes used in RAM are based on the codes devised by R.W.Hamming [10]. In the Hamming code, m parity bits are added to an n-bit data word,forming a new word of n+m bits. The position of bits are numbered in sequence from 1 to n +m. The position of the parity bits is

reserved with 2^k where k varies from 1 to n. The rest of bits are the data bits. The words of any length can be used with this code. Before describing the general characteristics of the Hamming code, we will represent its operation with eight bits of data word. Let's take an example the 8 bit data word

10110011. We include four parity bits with this word and arrange the 12 bits as follows:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	*P*3	*P*2	1	*P*1	0	1	1	*P*0	0	0	1	1

The 4 parity bits *P*3 *P*2 *P*1 and *P*0 are at positions 1, 2, 4, and 8 respectively. The 8 bits of data are at the rest of the positions. Each parity bit is calculated as follows:

*P*3 _ XOR of bits (3, 5, 7, 9, 11) = 1⊕0⊕1⊕0⊕1 = 1

*P*2 _ XOR of bits (3, 6, 7, 10, 11) = 1⊕1⊕1⊕0⊕1 = 0

*P*1 _ XOR of bits (5, 6, 7, 12) = 0⊕1⊕1⊕1 = 1

*P*0 _ XOR of bits (9, 10, 11, 12) = 0⊕0⊕1⊕1 = 0

As we all know that the odd function is performed by the exclusive-OR operation. It works as follow : value is equal to 1 for an odd number of 1's among the variables and value is equal to 0 for an even number of 1's. Hence each parity bit is set in such a way that the total number of 1's at the checked positions including the parity bit is always even. Flow graph for the transmitter section operation :

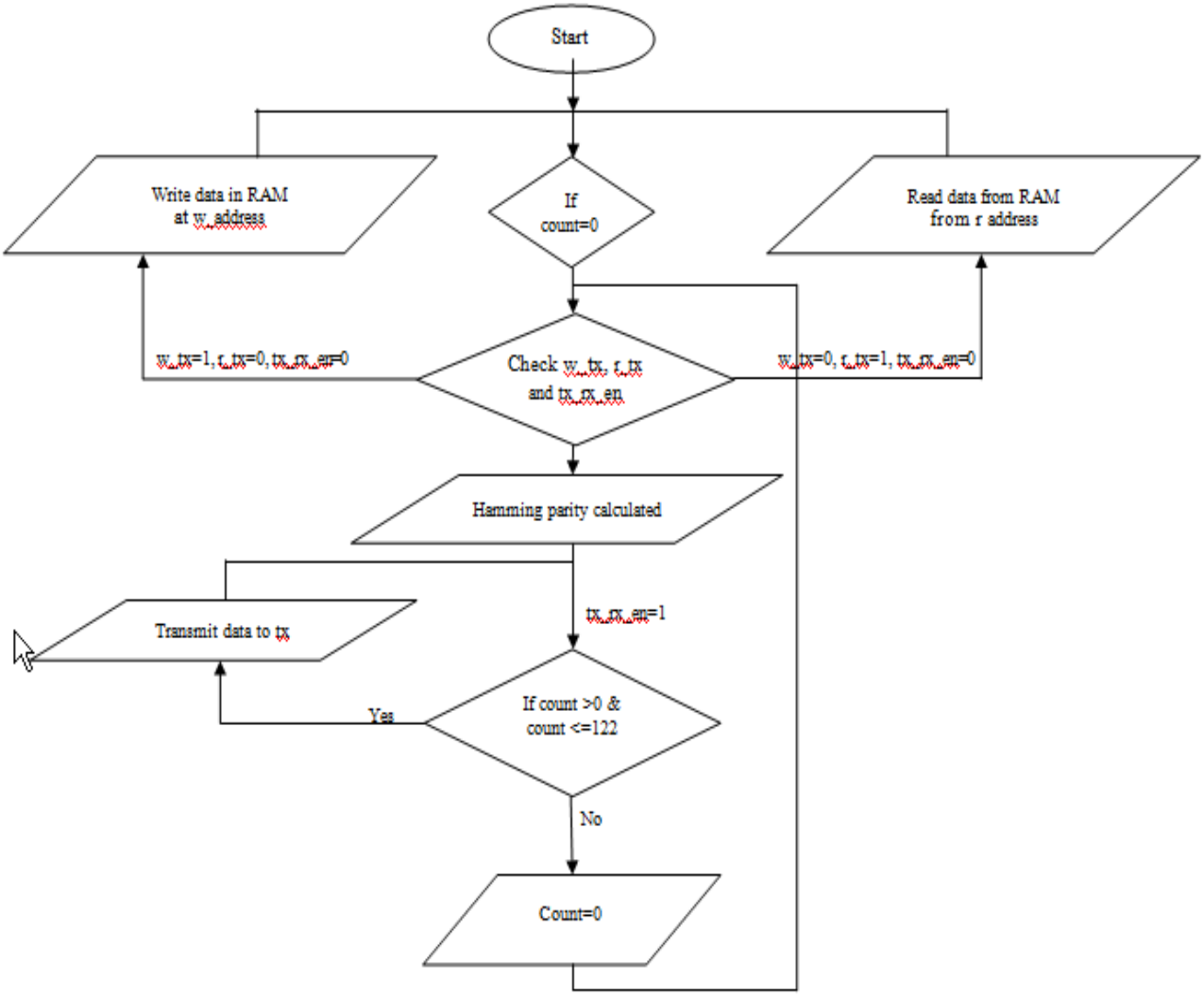

Fig 2. Transmitter Section Flow Graph

The 12 bit composite word which is to be written in the memory is composed of 8 bit data word along with the 4 bit parity. The 12 bit composite word which is written in the memory can be retained by simply putting 4 bit parity at the desired location:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	1	0	1	1	0	1	1	0	0	0	1	1

The rechecking of error is done when 12 bit data is read from memory.By considering a parity bits, the parity of data word is checked for the similar group of bits. The four check bits are calculated as follows:

*C*0 _ XOR of bits (1, 3, 5, 7, 9, 11)

*C*1 _ XOR of bits (2, 3, 6, 7, 10, 11)

*C*2 _ XOR of bits (4, 5, 6, 7, 12)

*C*3 _ XOR of bits (8, 9, 10, 11, 12)

From the above evaluation we get, 0 check bit represent an even parity and 1 check bit for an odd parity over the checked bits respectively. As the bits are written with even parity, the result, *C* _ *C*3*C*2*C*1*C*0 _0000, indicates that no error has occurred. However, if, the 4-bit binary number formed by the check bits gives the position of the erroneous bit if only a single bit is in error. For example, consider the following three cases:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
No Error	1	0	1	1	0	1	1	0	0	0	1	1
Error in bit 2	1	1	1	1	0	1	1	0	0	0	1	1
Error in bit 6	1	0	1	1	0	0	1	0	0	0	1	1

Here three cases arises

i) In the first case, the 12 bit word has no error.

ii) In the second case, there is an error in bit position number 2 because it toggle from 0 to 1.

iii) The third case shows an error in bit position 6 with a change from 1 to 0.

Performing the XOR operation on the corresponding bits, we examine the four check bits to be as follows:

	C_3	C_2	C_1	C_0
No error	0	0	0	0
Error in bit 2	0	0	1	0
Error in bit 6	0	1	1	0

Thus, We get *C* 0000, if we don't have any error; we obtain *C* 0010, with an error in bit 2; and with an error in bit 6, we get *C* 0110. It means, when *P* is not equal to 0, the decimal value of *C* shows the position of the bit in error. The error can then be corrected by toggle the corresponding bit. It is noticed that an error can occur in the data or in one of the parity bits. The Hamming code can be used for data words of any length. Generally for *m* check bits and *n* data bits, the total number of bits (*n* + *m)* that can be in a coded word is at most 2^m -1.

Only a single bit error can be detected and correction in the basic Hamming code. Although multiple bit errors can be detected but they are corrected erroneously, as if it consider them they were single bit errors. Further by adding of parity bit to the coded word, as a result of which the Hamming code can detect multiple bit error and correct single bit errors. The previous 12 bit coded word will becomes 101101100011P_{13} , where P_{13} is calculated from the XOR of the rest 12 bits when we include addition of this extra parity bit . This results in the formation of 13 bit word 1011011000111 having even parity. Whenever this word 1011011000111 is read from memory, the parity bit *P* and check bits are calculated for all the 13 bits. If value of *P* = 0, the parity is correct i.e even parity and if *P* = 1, the parity of the 13 bits is incorrect i.e odd parity. The following four cases can occur:

If $C = 0$ and $P = 0$ No error occurred.

If $C \neq 0$ and $P = 1$ A single error occurred that can be corrected.

If $C \neq 0$ and $P = 0$ A double error occurred that is detected but cannot be corrected.

If $C = 0$ and $P = 1$ An error occurred in the P_{13} bit.

This scheme will detect more than two fallacious bits in many cases, but is not guaranteed to detect all such errors. A modified Hamming code is used to generate and check parity bits for a single- error-correction, double-error-detection scheme in real systems. The modified code uses a different parity check bit scheme that balances the number of inputs to the logic for each check bit and thus the number of inputs to each circuit that does the checking. The balancing minimizes the delay through the error correction and detection circuits. These circuits can be used in a RAM subsystem to add check bits during write operations and to correct single errors and detect double errors during read operations.

As the calculated parity is stored in the parity storage ,The value of tx_rx_en signal is made high i.e equal to '1' to initialize the transmission of HDLC and for this the counter will run for the 0 to 122 count value for complete frame transmission. Starting from the MSB to LSB when all the 122 bits are transmitted, counter will reset automatically. It will raise the done signal to high value. Done signal is a status signal which gives the indication that the errors are corrected in all the registers after transmission.

The contents of an HDLC frame are shown in the table :

Table 1. Frame format for HDLC protocol

Start	Address	Control	Data Bits	FCS Or Parity bits	Stop
8 bits	8 bits	2 bits (00 By Default)	64 bits	32 bits	8 bits

3.2 Receiver Section

At the receiver side, initially condition tx_rx_en is checked, if its value is '1' then check for the value of counter. For starting the frame reception its value must be '0'. As all the conditions are met start the frame reception and start bits are received first. Now increment the counter value and get the receiver address from frame reception and verify the receive address check. According to the HDLC frame specify the 2 bit delay for control bit and later data part reception will start. When data part reception is done parity check or frame check sequence bits are received. At the end the reception of stop frame bits take place.

Perform the decoding of register data which is present in the HDLC frame and decode the received parity bits to detect the desired error location. Once the error is detected , perform the error correction mechanism at the detected address of the error. As the error correction is over RAM data is read from the corrected frame at the receiver and at this read data the RAM data write operation is performed as illustrated in the flow graph shown below :

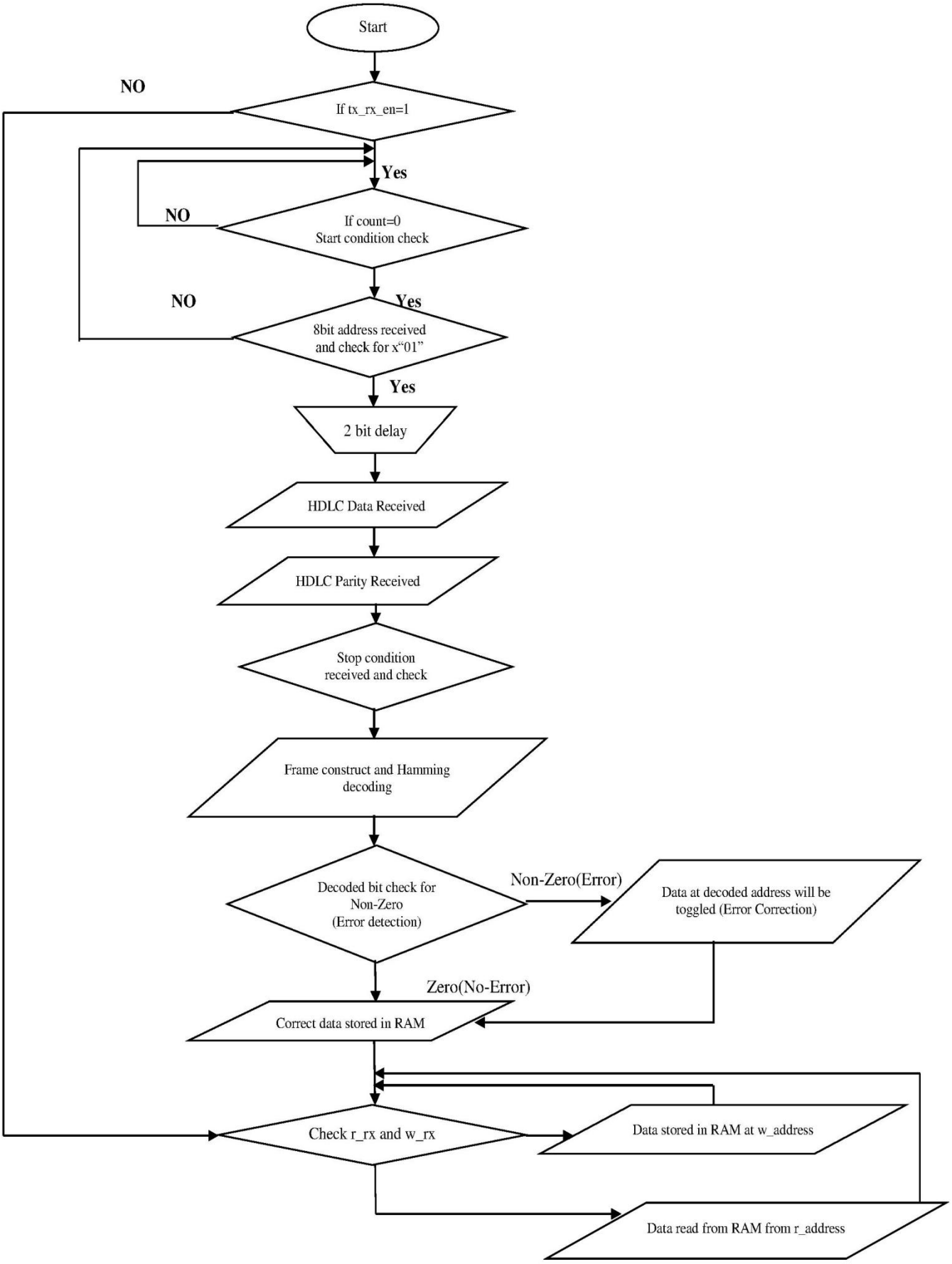

Fig. 3. Receiver section flow graph

4. RESULTS

In the proposed methodology, in accordance to the given algorithm errors are inserted in 8x8 matrix and it is observed that maximum number of errors that can be corrected are 8 bits. While maintaining the other performance parameters such as code rate and bit overhead. The resultant value that we obtained using equation (1,2) is given in the table II shown below.

The value for bit overhead and code rate are shown :

Table 2 : Result for Bit Overhead and Code Rate

S No.	Error correcting code	Data bits (m)	Redundant bits (n)	Length of code vector (k)	Bit overhead (B.O=n/m)	Code rate (R=m/k)	Error correction capability in No. Of bits
1	Proposed technique	64	32	96	50%	66.66%	8

The VHDL codes are written and simulated for 64 bit data. Simulation results for 12 bit coded word having 8 bit data and 4 bit parity is done. Eight bit errors are inserted and then these errors are corrected as shown in fig. 4 and fig. 5 respectively.

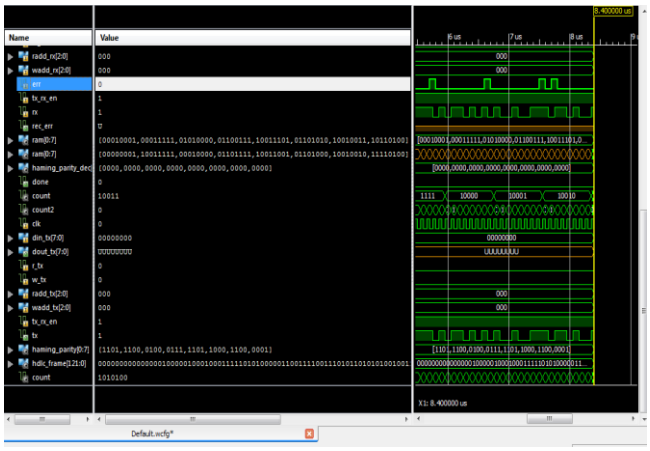

Figure. 4.

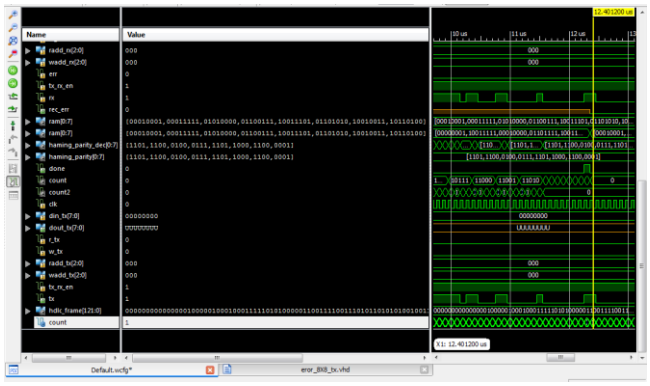

Figure. 5.

5. CONCLUSION

In this senate the error detection and correction is done by using hamming code in the proposed method. As this method is compared with other EDAC and it is found that the error correction capability of proposed method is eight error bit, encoder and decoder architecture has less complexity than other EDAC methods. The proposed technique can be used efficiently where retransmission of data is complicated and expensive too. The bit overhead and code rate for this technique are 50% and 66.66% respectively which provide better results for error detection and correction than other EDAC technique

6. ACKNOWLEDGMENT

I would like to place on my record my deep sense of gratitude to my teacher Mr. Narinder Sharma for his guidance, help and constructive suggestions. He has been extremely helpful and has aided for all the material essential for the preparation of this work. I express my sincere gratitude to Dr. V. K. Banga, principal- A.C.E.T, for their stimulating guidance and continuous encouragement. I would also like to thank my family, colleagues and the almighty for showing me the right direction to carry out my research work.

7. REFERENCES

[1] Anlei Wang, Naima Kaabouch, "FPGA Based Design of a Novel Enhanced Error Detection and Correction Technique, IEEE, Vol. 3, Issue No. 5,March 2008, pp 25-29.

[2] Berlekamp, E.R. , Algebraic Coding Theory, McGraw-Hill, New York,1968.

[3] Berrou, Glavieu, Thitimajshima, *"Shannon limit error-correcting coding and decoding: Turbo-codes*" International Conference on Communications, 1993, pp 1064-1069.

[4] Behrouz A. Forouzan *"Data Communication and networking"* 2nd edit. Tata McGraw Hill.

[5] Brajesh Kumar Gupta, " 30 BIT Hamming Code for Error Detection and Correction with Even Parity and Odd Parity Check Method by using VHDL", International Journal of Computer Applications, Vol. 35, Issue No.13, December 2011, pp 31-38.

[6] B. K. Gupta, R. L. Dua, "*Review Paper on Communication by Hamming Code Methodologies*", International Journal of Electrical, Electronics and Computer Engineering, 2011, pp. 52-54.

[7] Fernanda Lima, Luigi Carro, Ricardo Reis *"Designing Fault Tolerant Systems into SRAM-based FPGAs"* Anaheim,Vol. 3, June 2003,pp 312-318**.**

[8] Gaurav Chandil, Priyanka Mishra "*Study and Performance Evaluation of Xilinx HDLC Controller and FCS Calculator*" IOSR Journal of Engineering (IOSRJEN), Vol. 2, Issue 10, October 2012, pp 41-50.

[9] Heesung Lee, Joonkyung Sung, and Euntai Kim, *"Reducing Power in Error Correcting Code using Genetic Algorithm",* World Academy of Science, Engineering and Technology 25 2007.

[10] HAMMING, R. W., "*Error Detecting and Error Correcting Codes*", Bell System Tech. Jour.,Vol. 29, 1950, pp 147–160.

[11] Golay (23,12,7) code - Error Correcting Codes Page http://www.eccpage.com/golay23.c

[12] Golay Code: http://mathworld.wolfram.com/GolayCode .html

[13] Morris Rubinoff, "*N Dimensional Codes for Detecting and Correcting Multiple Errors*", Communication of ACM,Vol. 4, Issue No. 12,Dec 1961,pp 545-551.

[14] M. Pflanz, K. Walther; C. Galke, H.T. Vierhaus, "*On-Line Techniques for Error Detection And Correction in Processor Registers With Cross-Parity Check*", Journal of Electronic Testing , Vol. 19, Issue No. 5, April 2003, pp 501-510.

[15] M. Imran , Z. Al-Ars, G. N. Gaydadjiev, "*Improving Soft Error Correction Capability of 4-D Parity Codes*", IEEE transaction, Vol. 4, Issue No. 2, January 2009, pp 233-238.

[16] M.Kishani, H. R. Zarandi, H. Pedram, A Tajary, M. Raji, B. Ghavami, "*Horizontal-Vertical Diagonal Error Detecting and Correcting Code to Protect Against With Soft Errors*" Springer science, Vol. 15, Issue No. 3-4, May 2011, pp 289-310.

[17] Narinder Pal Singh, Sukhjit Singh, Vikrant Sharma, Amandeep Sehmby, "*Ram Error Detection And Correction Using HVD Implementation"* European Scientific Journal,Vol. 9, Issue No.33, November 2013,pp 424-435.

[18] P. Sharma, K. Kumar, A. K. Singh, "*Single Bit Error Detection And Correction by Matrix Method*", International Journal of Research & Innovation in Computer Engineering, Vol. 2, Issue No. 2, August 2012, pp. 201-206.

[19] S.Lin and Jr. D.J. Costello, " Error control coding: fundamentals and applications", 1983 by Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632, ISBN 0-13-283796-X

[20] S. Sharma, Vijay Kumar, " *An HVD Based Error Detection and Correction of Soft Errors in Semiconductor Memories Used for Space Application*", International conference on devices, circuits and systems (ICDCS), March 2012, pp. 563-56.

[21] Shubham Fadnavis, "*An HVD Based Error Detection And Correction Code In HDLC Protocol Used For Communication",* International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue No.6, June 2013, pp 2349-2353.

[22] T. Kasarni , A Kitai and S. Lin "*On the Undetected Error Probability for Shortened Hamming Codes*", IEEE Transaction Communication , Vol.33, Issue No. 2, 1985, pp 570 -574.

[23] T. Fujiwara et al., "*Error Detecting Capabilities of the Shortened Hamming Codes Adopted for Error Detection in IEEE Standard 802.3*," IEEE, Vol. 37, Issue 9, September 1989, pp 986-989.

[24] T.Fill and P Glenn Gulak , " An assessment of VLSI and embedded software implementations for Reed-Solomon decoders,IEEE Signal Processing Systems SIPS ", 02 Oct 2002,pp 99-102.

[25] Vishal Badole, Amit Udawat, "*Implementation of Multidirectional Parity Check Code Using Hamming Code for Error Detection And Correction",* International Journal of Research in Advent Technology, Vol.2, Issue No.5, May 2014, pp 1-6.

[26] Togneri R and deSilva CJS , " Fundamentals of information theory and coding design, discrete mathematics and its applications", CRC Press, New York,2002 ISBN 1-58488-310-3.

[27] Y. Bentoutou, "Program Memories Error Detection and Correction On- Board Earth Observation Satellites", World Academy of sciece, Engineering and Technology 66 2010.

[28] Zainalabedin Navabi, *"VHDL Modular Design and Synthesis of Cores And Systems"* 3rd edition ,Tata McGRAW-Hill private limited,New Delhi, 2011.