FPGA Implementation of Fast Error Correction and Detection for Memories

Kunjan D. Shinde
2nd year M.Tech in Digital Electronics
Department of Electronics &Communication Engineering.,
S.D.M College of Engineering & Technology, Dharwad-02, Karnataka, India.

Jayashree C. Nidagundi
Assistant Professor , Department of Electronics &Communication Engineering.,
S.D.M College of Engineering & Technology, Dharwad-02, Karnataka, India.

Dr. S.R.Patil
Dean & HOD, Department of Electronics &Communication Engineering.,
PDA College of Engineering & Technology, Gulbarga, Karnataka, India.

Abstract— The higher integration technologies made it possible for accessing any device so fast that within a fraction of seconds the job can be performed. Now days fast memories exists everywhere during accessing if any error happens that has to be detected and corrected within a fraction of microseconds that is made possible with help high performance error correcting codes(ECCs) such as LDPC and Turbo Codes. The proposed paper deals with coding and decoding of EG-LDPC codes using majority logic decoding mechanisms. The new control logic is developed for decoding so that error correction can be possible within 3 cycles if the transmitted code vector is error free. The implementation result shows that fast error correction prototype is possible with lower area and low power.

Keywords- Error Correcting Codes(ECCs), Euclidean geometry low density parity check codes(EG-LDPC), Majority logic(ML) codes, Field programmable gate array(FPGA),Verilog

I. INTRODUCTION

The reliability of memories depends on dimensions of the circuit, operating voltages, and integrated density Triple modular redundancy codes and error correcting codes are mostly used for error correcting. In those techniques a majority vote has been used to gives the perfect output and triggering the error correcting mechanism respectively. But they have high power consumption due to the voter circuit and it requires large area. Generally there is single bit error, double bit errors and multiple bit error detections codes are normally present in digital circuits. The decoding and encoding are so flexible in those types of codes. The consequence of augmenting integration densities, it increases the number of soft errors, which needs higher error correction capabilities.

Some multi error bit corrections Codes are reed Solomon codes and Bose- Chaudhuri- Hocquenghem codes, but in which the algorithm is so complex and it is iteration based. The decoders to decode in fixed rate and so it reduces the operating criteria. To attain higher capability to detect errors among error correcting codes the sub group of low density parity check code called quasi cyclic LDPC codes has been selected to increase the performance of the decoders to detect and correct large number of errors. It belongs to family of majority logic decoding. The reason for using ML decoding is that it is very simple to implement and very practical and has low complexity.

Another alternative is to first detect if there are errors in the word and only perform the rest of the decoding process when there are errors. This greatly reduces the average power consumption as most words will have no errors. Error detection in a block code can also be implemented by computing the syndrome and checking whether all its bits are zero. By calculating the syndrome, we can implement a fault detector for an ECC is but this also would add an additional complex functional unit. This paper focus on using the MLD circuitry itself as an error detecting module therefore with no additional hardware the read operations could be accelerated.

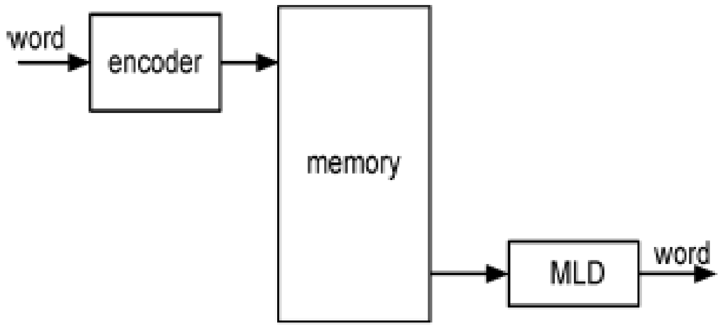


Figure 1 General Memory System

The parallel encoders and decoders have been implemented to overcome the drawback of majority logic decoder in which it takes N number of cycles to detect. In this paper we are detecting error in memory device itself so the data corruption during processing has been eliminated easily to give improved performance. The general memory system implemented with majority logic decoder as shown in the figure 1.

The remainder of this paper is organized as follows. In Section II, a brief overview of Encoder construction and Decoding using Majority logic is discussed. Implementation details of Encoder and Decoder design is discussed in Section III. Section IV gives discussion of experimental results.

II. OVERVIW OF CODING AND DECODING

LOW-DENSITY parity-check (LDPC) codes, discovered by Gallager in 1962 [1], were rediscovered and shown to approach Shannon capacity in the late 1990s[2]. LDPC codes are subclass of linear block codes. Today LDPC codes are being considered for a wide variety of emerging applications such as high-density flash memory, satellite broadcasting, WiFi and mobile WiMAX. Rapid performance evaluation of LDPC codes is very desirable to design better codes for these applications. LDPC codes are decoded iteratively using the message passing and minimum sum [1], [2] Due to the inherent parallelism decoding speed is very high. Hence, a complex interconnect network is required which consumes a significant amount of silicon area and power.

A. Encoder Construction

Given a generator polynomial $g(x)=1+X^4+X^6+X^7+X^8$ the encoder(15,7,5) can be constructed. The encoded vector mainly consists of two parts, the first part consist of information bits and second part is the parity bits, where each parity bit is simply an inner product of information vector and a column of X , from G=[I:X][3,4] as shown in fig.2.

$$\begin{array}{c|ccccccc|cccccccc|}
 & c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} \\
i_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
i_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
i_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
i_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
i_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
i_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
i_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
 & \multicolumn{7}{c}{\underbrace{\hspace{10em}}_{I}} & \multicolumn{8}{c}{\underbrace{\hspace{12em}}_{X}}
\end{array}$$

Figure 2: Generator matrix for (15,7,5)EG-LDPC codes

The encoder circuit [6] to compute the parity bits of the (15, 7, 5) EG-LDPC code is shown in Fig3. In this figure, the information vectors are (i0,….i6) and will be copied to (c0,..,c6) bits of the encoded vector, c. The rest of encoded vector (c7…c14), that is the parity bits are the linear sums (XOR) of the information bits.

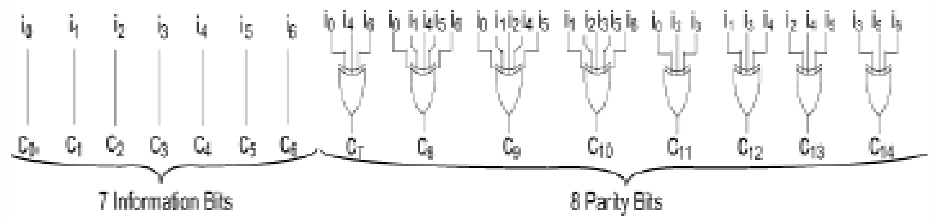


Figure 3 Encoder circuit of (15,7,5)EG-LDPC Codes

B. Decoding Logic

Among the error detection and correction technique one step majority correction is a fast and relatively efficient with low complexity error-correcting technique [5]. One-step-majority correctable ECC codes are limited which include type-I two-dimensional EG-LDPC. The majority logic decoder have the method of working in which from the received codeword itself the correct values of each bit under decoding can directly found out. This method consists of mainly two steps-

1) Generating a specific set of linear sums of the received vector bits using the xor matrix 2) Determining the majority value of the computed linear sums. It is the majority logic output which determines the correctness of the bit under decoding. If the majority output is ‘1’, then the bit is inverted, otherwise would be kept unchanged.

As described before, the ML decoder is powerful and simple decoder, which has the capability of correcting multiple random bit-flips depending on the number of parity check equations. It consists of four parts: 1) a cyclic shift register; 2) an XOR matrix; 3) a majority gate; and 4) an XOR for correcting the codeword bit under decoding. The circuit of serial one-step majority logic corrector [5]] for (15, 7, 5) EG-LDPC code is shown in Fig.4. The cyclic shift register is initially stored with the input signal x and shifted through all the taps. The results {Bj} of the check sum equations from the XOR matrix is calculated from the intermediate values in each tap. In the Nth cycle, the result would reach the final tap, producing the output signal, which is decoded version of the input.

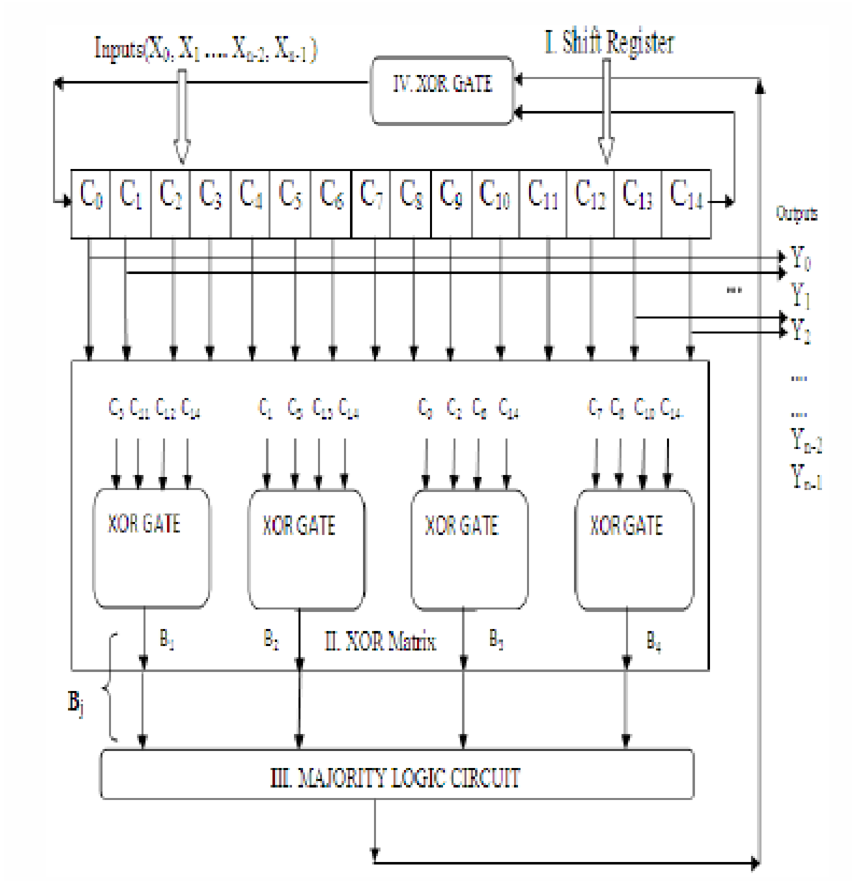

Inputs(X_0, X_1 .... X_{n-2}, X_{n-1})
IV. XOR GATE
I. Shift Register
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14}
Outputs
Y_0
Y_1
Y_2
Y_{n-2}
Y_{n-1}
C_3 C_{11} C_{12} C_{14}
C_1 C_5 C_{13} C_{14}
C_0 C_2 C_6 C_{14}
C_7 C_8 C_{10} C_{14}
XOR GATE
XOR GATE
XOR GATE
XOR GATE
B_1
B_2
II. XOR Matrix
B_3
B_4
B_j
III. MAJORITY LOGIC CIRCUIT

Figure 4 Type-I Majority Logic Decoder

III. IMPLEMENTATION DETAILS OF ENCODER AND DECODER

With a given generator polynomial, $g(x)=1+X^4+X^6+X^7+X^8$ the parity check matrix is generated and encoder circuit is implemented using Verilog HDL programming language. The Type-I majority logic decoding is used as explained in section II with slight changes in the decoding procedure to reduce the N number of decoding cycles. Our proposed scheme uses the decoding procedure same as that of above for first 3 cycles, with designed control logic which tests the output register after 3rdcycle if output register is zero then there are NO errors and decoding stops at the 3rd step only otherwise it takes the 15 cycles to correct the error. The procedure is explained in the form of flowchart as shown below in fig.5.

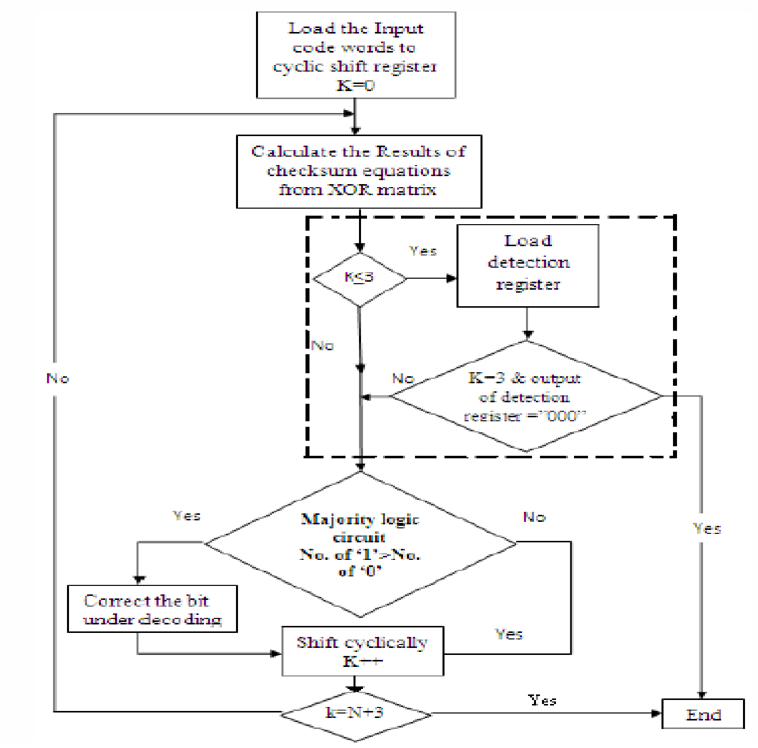

Load the Input code words to cyclic shift register K=0
Calculate the Results of checksum equations from XOR matrix
K≤3
Yes
Load detection register
No
K=3 & output of detection register ="000"
No
No
Yes
Majority logic circuit No. of '1'>No. of '0'
Yes
No
Correct the bit under decoding
Shift cyclically K++
Yes
k=N+3
Yes
End

Figure 5 Flowchart of Majority logic decoding with control logic

The ML decoder works with a 15-tap shift register, an XOR array to calculate the orthogonal parity check sums and a majority logic circuit which will decide whether the current bit under decoding is erroneous and the need for its inversion. The plain ML decoder [6] shown in Figure 4 is also having the same schematic structure up to this stage. The additional hardware intended for fault detection illustrated in Fig.6 are: a) the control logic unit and b) the output tristate buffers. The control unit triggers a finish flag when there is no errors are detected in data read. The output tristate buffers are always in high impedance state until the control unit sends the finish signal so that the current values are forwarded to the output y from the shift register. The detection process is managed by the control unit[7]. For distinguishing the first three iterations of the ML decoding, a counter is used here which counts up to three cycles. The control unit evaluates the output from xor matrix Bj by giving it as input to the OR 1 gate. This output value is fed to two shift registers which has the results of the previous stages stored in it. The values are shifted accordingly. The third coming input is directly forwarded to the OR 2 gate and finally all are evaluated in the third cycle in the OR 2 gate. If the result is "0," a finish signal is send by the FSM which indicates that the processed word is error-free. The ML decoding process runs until the end, if the result is "1".

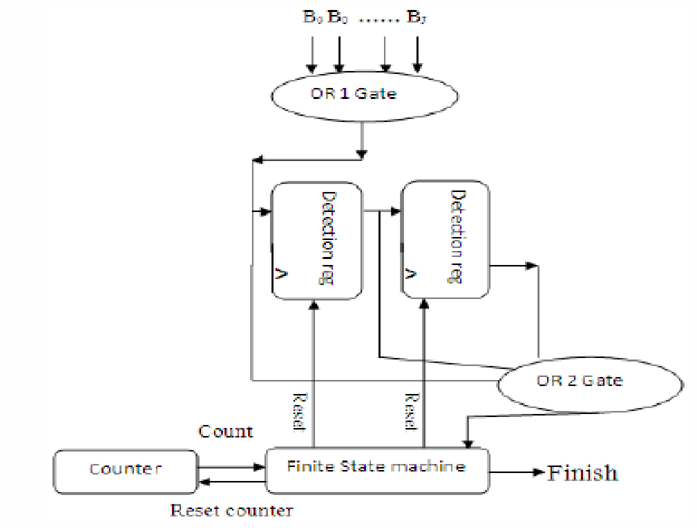


Figure6 Control Logic.

The majority logic gate decoding is implemented by using verilog . That is two level logic [6].If during the memory read access an error is detected, the xor gate will correct it, by inverting the current bit under decoding. The EG LDPC code used here is only for 15 bits, it have only outputs four outputs from xor matrix. The orthogonal majority parameters B1,B2,…BN are constructed using sorting networks. This clearly provides a performance improvement respect to the traditional method .The proposed method mostly would only take three cycles for decoding(five, if we consider the other two for input/output) since most of the words would be error free and would need to perform the whole decoding process only for those words with errors (which should be a minority).

IV EXPERIMENTAL RESULTS

In this section the simulations results of the proposed Majority Logic Decoder/Detector and the encoder is presented. The front end design of the architecture, its simulation, synthesis and comparison are done using XILINX ISE Design Suite 7.1. The target device is Spartan 3E-XC3S400. The designs are coded in Verilog HDL language. A codeword of size 15 is chosen here for designing.
The proposed majority logic decoder and encoder techniques are simulated for both error free and erroneous conditions the during memory access, and the results are shown below in figure 8,9 and 10.

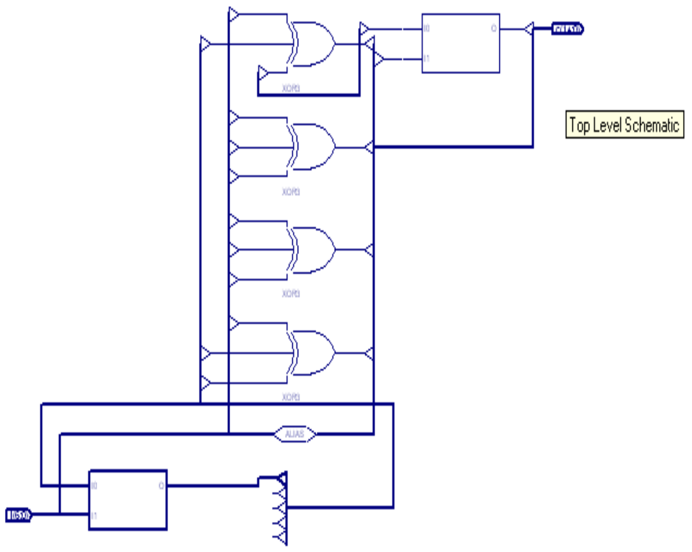


Figure8 RTL schematic of Encoder

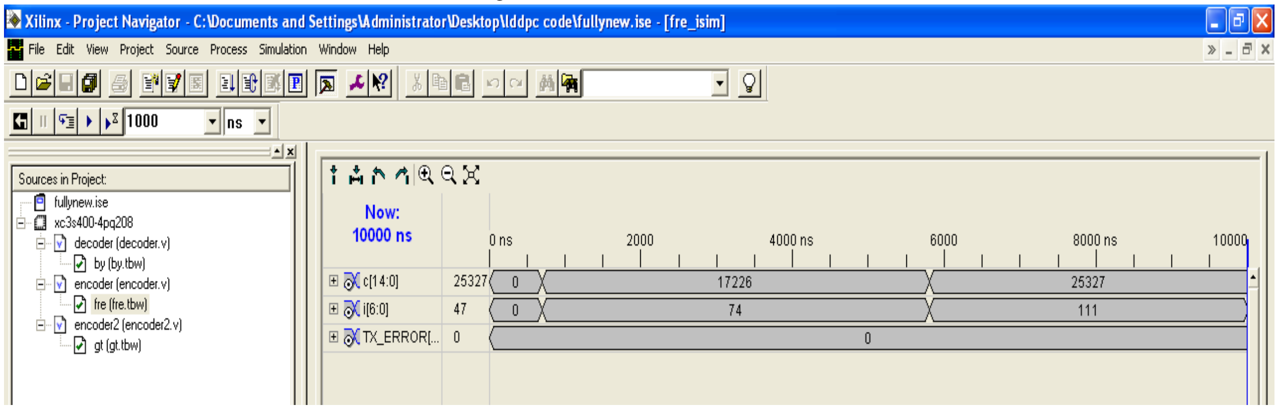
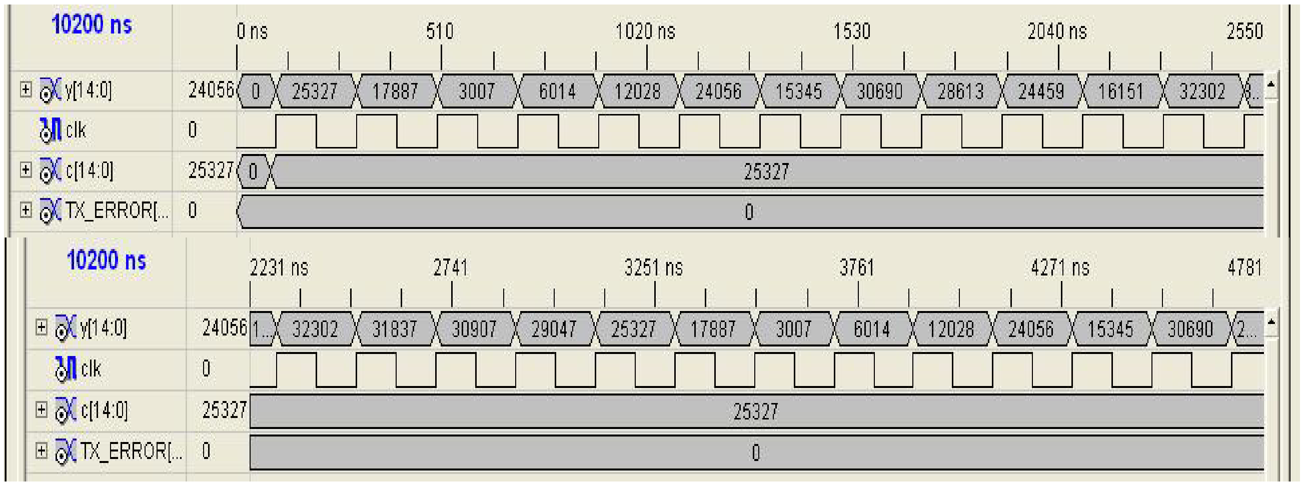

Xilinx - Project Navigator - C:\Documents and Settings\Administrator\Desktop\ldpc code\fullynew.ise - [fre_isim]
File Edit View Project Source Process Simulation Window Help
1000 ns
Sources in Project:
fullynew.ise
xc3s400-4pq208
decoder (decoder.v)
by (by.tbw)
encoder (encoder.v)
fre (fre.tbw)
encoder2 (encoder2.v)
gt (gt.tbw)
Now: 10000 ns
0 ns 2000 4000 ns 6000 8000 ns 10000
c[14:0] 25327 0 17226 25327
i[6:0] 47 0 74 111
TX_ERROR[... 0 0

Figure9 Encoder Simulation Results



Decoder Simulation Results with cycle time equal to codeword length N= 15

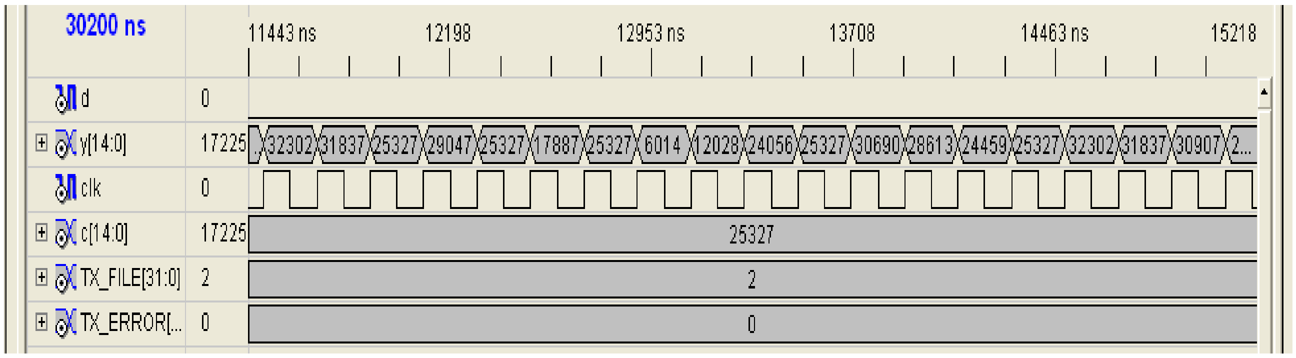
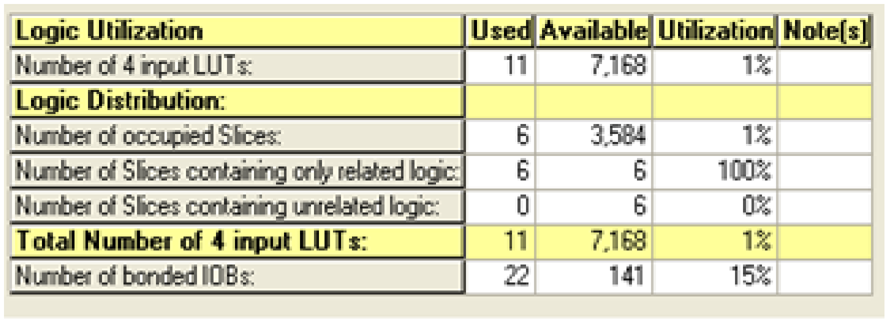


Figure 10 Decoder Simulation Results with cycle time N =3



Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs:	11	7,168	1%	
Logic Distribution:				
Number of occupied Slices:	6	3,584	1%	
Number of Slices containing only related logic:	6	6	100%	
Number of Slices containing unrelated logic:	0	6	0%	
Total Number of 4 input LUTs:	11	7,168	1%	
Number of bonded IOBs:	22	141	15%	

Table 1 Encoder Implementation Results

V. CONCLUSION

The paper focuses on the design of Encoder and Majority Logic Decoder/Detector (MLDD) for fault detection along with correction of fault, suitable for memory applications, with reduced fault detection time. From the results, (A codeword of size 15 is chosen here for designing), when compared to the normal MLD, The proposed MLDD has comparatively less delay of 12.578 ns and can detect the presence of errors in just 3 cycles even for multiple bit flips.

It has found that for error detection and correction (for codeword of 15), when comparing to the existing technique, a speed up of about 1100 ns is obtained when there is no errors in data read access. It's because the fault detection needs only three cycles and after the detection of an error free condition, the codeword is passed to the output without further corrections. This is a great saving of time since most of the situations the memory read access does not make errors. Therefore there is a considerable reduction in the memory access time.

MLDD error detector is designed as it is independent of the code word size and inference about area is that for large values of code word size, the area overhead of the MLDD actually decreases with respect to the plain MLD technique. i.e., for large values of code word size both areas are practically the same. Therefore the proposed work will be an efficient design for fault detection and correction.

.

VI. ACKNOWLEDGMENTS

We thank the Management, the Principal/Director and Staff of S.D.M College of Engineering and Technology, Dhavalgiri, Dharawad, Karnataka, India for encouraging us for this research work.

REFERENCES

[1] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inf.Theory, vol. IT-8, pp. 21–28, Jan. 1962.
[2] D. J. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," Electron Lett., vol. 32, pp. 1645–1646,1996.

[3] R.C.Baumann,"Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Reliabil.*, vol. 5, no.3, pp. 301–316, Sep. 2005.

[4] C. W. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," *IEEE* Tr*ans. Device Mater. Reliabil.*, vol. 5, no. 3, pp. 397–404, Sep. 2005.

[5] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for NanoMemory applications," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 17, no. 4, pp. 473–486, Apr. 2009.

[6] Shih-Fu Liu,Pedro Revingo, and Juan Antonio Meastro "Efficient majority fault detection with difference set codes for memmory applications", *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.,* vol. 20, no. 1, pp. 148–156, Jan. 2012.

[7] R.Meenaakshi Sundhari1, C.Sundarrasu2, M.Karthikkumar3 "An Efficient Majority Logic Fault Detection to reduce the Accessing time for Memory Applications", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013.

[8] Shu Lin &Daniel J. Costello " Error Control Coding 2ndedition.