IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FPGA based Design and implementation of Triple Error Detection and Correction Algorithm using BCH Codes

J. Balakrishna

Department of ECE, CBIT(A), Hyderabad, India

***Abstract*— Data corruption during the transmission and reception of data because of noisy channel medium is the most common problem faced in the digital communication system. Thus, it is hard to get reliable communication. To get error-free communication we need Error correction code. Hamming codes, Convolution Codes, Reed Mueller codes, Turbo codes and BCH codes are some of the Error Correcting Codes.**

BCH stands for Bose, Ray – Chaudhuri, Hocquenghem, BCH codes are powerful class of multiple error correction codes with well-defined mathematical properties. The mathematical properties of BCH codes are defined as the Galois Field or Finite Field Theory. The focus is to design triple error correcting BCH decoder architecture.

Syndrome Generator, Error Counter, Error Corrector, Flip-Flops, Multiplexer collective design makes BCH Decoder. The BCH codes architecture is described using hardware description language called Verilog and synthesized using Xilinx ISE.

***Keywords*— BCH codes, Syndrome Generator, BCH decoder, Error counter, Error corrector**

I. INTRODUCTION

In the Present Digital Communication systems, there is a lot of demand for reliable data transmission. The environmental interference and the physical defects in the medium are the major causes of the data or message corruption in the communication medium, which leads to the insertion of random bits into the original message and alter the original message. To have a consistent communication through a noisy medium that has an unacceptable Bit Error Rate (BER) and low Signal to Noise Ratio (SNR), we need to have Error Correcting Codes (ECC).

There are many types of error correction codes are used in the present digital communication system are based on the type of error expected, the communication medium expected error rate, and weather re-transmission is possible or not. Some of the error correction codes, which are widely used these days, are BCH, Turbo, Reed Solomon, LDPC, and Hamming codes. These codes are diverse from each other in their complexity and implementation. One of the simplest and linear block codes are Hamming codes. They are proficient of correcting only one arbitrary error and therefore are basically not useful unless a simple error control circuit is required.

The best classic error-correcting codes are the Bose, Chaudhuri, and Hocquenghem (BCH) codes that are a generalization of the Hamming codes for multiple-error correction. The BCH codes form a large class of powerful random error-correcting cyclic codes having capable of multiple error correction. BCH codes function over finite fields or Galois fields

Error correction is taken place by accumulating parity bits to the original message bits during transmission of the information or data. The addition of parity bits to original data makes the size of the original message bits longer. Now this longer message bits are called “Codeword”. This codeword is received by the receiver at the destination and could be decoded to retrieve the original message bits. Error-correcting codes are used in most of the digital applications, space and satellite communication, compact disk players, 2-Dimensional Barcodes, and cellular telephone networks.

This paper is organized as follows; Section II emphasizes on literature work. Section III briefly deals with design and architecture of BCH Decoder. Section IV focuses on design of triple error detection and correction (TED-TEC). Simulation and synthesis results of bch decoder are discussed in Section V. This work is concluded in Section VI.

II. LITERATURE REVIEW

Two parallel implementations of Double Error Correcting (DEC) BCH codes have been considered in [1]. The first implementation of a DEC BCH code (DEC BCH) is a standard parallel Implementation. The Second implementation of DEC BCH codes (DEC BCH II) is by removing all sub circuits and gates which solely contribute to the correction of 3-bit errors. The parallel decoding algorithm was developed and implemented using the Synopsis Design Compiler and a 130-nm industrial standard cell library.

Different coding methods detect and correct from single-bit error to multi-bit or burst errors with the corresponding penalty in code rate, bit overhead, area, and power consumption. Hamming code is well known for its single-bit error detection & correction capability. To provide such a capability, it introduces 4 redundancy bits in a 7-bit data item. The redundancy bits will be appended at the end of the data bits. This eliminates the overhead of the redundancy bits at the sender end and their removal at the receiver end after checking for single-bit error and consequent correction. The method proposed in [2] is highly scalable without the

overhead. Because of this feature, it is suitable for the transmission of large size data bit-streams with much lower redundancy bits per data bit ratio.

Existing codes for reducing Multiple Bit Upsets (MBUs) mainly focus on the correction of up to 3-bit burst errors. As the technology scales and cell interval distance decrease, the number of affected bits can easily extend to more than 3 bits. A technique to extend 3-bit Burst Error-Correction (BEC) codes with Quadruple Adjacent Error Correction (QAEC) is presented in [3]. They do not require additional parity check bits compared with a 3-bit BEC code. The best solutions are presented for 16 data bits and the best solutions found in reasonable computation time are presented for 32 and 64 data bits.

In [4], error detection and correction technique based on the Orthogonal Codes Convolution, Closest Match, and vertical parity is discussed. Orthogonal codes are binary values, and they have an equal number of 1's and 0's. There are 8 orthogonal codes and 8 antipodal codes for a total of 16 bi-orthogonal codes. Antipodal codes are just the inverse of orthogonal codes. The transmitter does not have to send the parity bit for the code, if there is a transmission error, the receiver may be able to detect it by generating a parity bit at the receiving end.

A double-error-correcting and triple error- detecting (DEC-TED) Bose–Chaudhuri–Hocquenghem (BCH) code decoder is presented in [5]. To increase the decoding efficiency, an adaptive error correction technique is presented that detects the number of errors in a codeword immediately after syndrome generation and applies a different error correction algorithm depending on the error conditions, with the adaptive error correction technique, the average decoding latency and power consumption are reduced.

III. Design and Architecture of BCH Decoder

The architecture and design for decoding data or information using Binary BCH codes are discussed in this section. The BCH decoder design is shown in Figure 1.

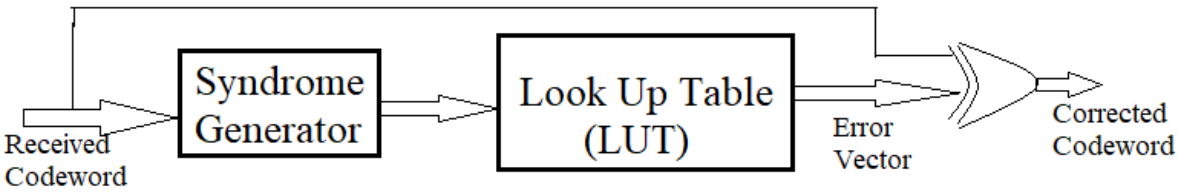


Fig. 1: Architecture of BCH Decoder.

The BCH decoder has the following two modules:

Syndrome Generator

Look-Up Table (LUT)

The implementation and the algorithms used to design the above said modules vary with the architectures.

A. Syndrome Generator

The syndrome calculator is the first module at the decoder also, the design of this module is almost same for all the BCH code decoder architecture. The input to this module is corrupted codeword. The equations for the codeword, received bits and the error bits are given in equations 1,2 and 3.

Codeword equation $c(x) = c_0 + c_1x + c_2x^2 + ... + c_{n-1}x^{n-1}$ (1)

Received bits equation $r(x) = r_0 + r_1x + r_2x^2 + ... + r_{n-1}x^{n-1}$ (2)

Error bits equation $e(x) = e_0 + e_1x + e_2x^2 + ... + e_{n-1}x^{n-1}$ (3)

Thus, the final transmitted data polynomial equation is given in the equation 4.

$$r(x) = c(x) + e(x) \quad (4)$$

The first step at the decoding process is to store the transmitted data polynomial in the buffer register and then to calculate the syndromes S_j. The important characteristic of the syndromes is that depends on only error location, not on the transmitted information. The equations of the syndromes are given as follows:

Define the syndromes S_j as

$$S_j = \sum_{i=0}^{n-1} r_i \alpha^{i \cdot j} \quad for \ (1 \le j \le 2t).$$

Since $r_j = c_j + e_j$ (j = 0, 1, ..., n-1)

Rewrite the syndrome equation as:

$$S_j = \sum_{i=0}^{n-1} (c_i + e_i)\alpha^{i \cdot j} = \sum_{i=0}^{n-1} c_i \alpha^{i \cdot j} + \sum_{i=0}^{n-1} e_i \alpha^{i \cdot j}$$

By the definition of BCH codes

$$\sum_{i=0}^{n-1} c_i \alpha^{i \cdot j} = 0 \quad for \ (1 \le j \le 2t).$$

Thus,

$$S_j = \sum_{i=0}^{n-1} e_i \alpha^{i \cdot j} \quad (5)$$

The equation (5) indicates the output of the syndrome calculator. From the equation, it can be observed that the syndromes depend on only error polynomial e(x), so if there is no error occurs during the transmission then all the generated syndromes will be zero.

B. Look-Up Table (LUT)

The LUT[1] contains all the possible pairs of syndromes and their corresponding error patterns. In this decoder, the error positions can be determined directly from the syndromes after a syndrome vector is computed.

In the LUT-based decoder, the error vector can be directly determined immediately after the syndrome vector is computed. Thus, the LUT-based decoder has a shorter decoding latency, but the area increases.

C. Computing Syndrome

The parity-check matrix H of (n, k, t) DEC-TED BCH code is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(n-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{H_1} \\ \mathbf{H_3} \end{bmatrix}$$

where K is the number of information bits, n is the length of the Block code, α is the primitive element, t is the no of errors in GF(2m).

To determine whether the received codeword 'v', has errors, syndrome vector 'S' is calculated as

$$\mathbf{S} = \mathbf{v} \cdot \mathbf{H}^{\mathrm{T}} = \left[\mathbf{v} \cdot \mathbf{1}^{\mathrm{T}}, \mathbf{v} \cdot \mathbf{H}_1^{\mathrm{T}}, \mathbf{v} \cdot \mathbf{H}_3^{\mathrm{T}}\right] = [S_0, S_1, S_3]$$

where S_0 is a 1-bit vector, S1 and S3 are m-bit vectors for the code generated in GF(2m). A single-bit error can be corrected using only the S1 vector as H1 can be used as the parity-check matrix for the Hamming code.

For a double bit error correction, S_1 and S_3 vectors are utilized together.

The Error is corrected using the XOR gate, the inputs for the XOR gates are Error vector obtained from corresponding Syndrome and the received codeword.

Table 1: Syndrome Condition for number of errors

Number of Errors	Syndrome Condition	
No Error	S_0	S_1 and S_3
No Error	0	$S_1=S_3=0$
Single-bit Error	1	$S_1^3=S_3$
Double-bit Error	0	$S_1^3 \neq S_3$
Triple-bit Error	0	$S_1^3 \neq s_3$

D. Dynamic Power Problem in Fully Parallel BCH Decoders

Previous studies on a fully parallel architecture for the BCH decoder have focused on circuit optimization methods to reduce the latency while minimizing the complexity of the implementation. However, considering that the read, write a power of emerging memories is generally at the microwatt level, much higher power consumption in conventional fully parallel decoders undermines the low-power advantage of emerging memories.

The high dynamic power consumption in the fully parallel BCH decoders can mainly be attributed to the syndrome vector dependence in decoder blocks following the syndrome generator.

The syndrome vector is a key factor in finding errors in LUT-based decoders. Whenever a new input is entered into the decoder, the syndrome generator produces invalid glitches before its outputs settle down. The glitches cause undesired transitions at internal nodes in the blocks that follow the syndrome generator LUT-based decoder and increase dynamic power consumption.

E. Adaptive Error Correction Technique for Double Error Correction

The implementation of stable syndrome vector using adaptive error correction and an invalid transition inhibition technique is proposed to achieve high decoding efficiency and low-power consumption.

After syndrome vectors are generated, the number of errors caused in the received codeword is classified in an error counter block, and a 2-bit flag signal that represents the number of errors is generated. Then, different error correction algorithms are applied depending on the generated flag signal to improve the decoding efficiency, and a proper error vector is added to the received codeword through the 3:1 MUX.

The 2-bit flag signal can be generated based on the generated syndrome vectors. For odd numbers of errors (single- or triple-bit errors), S0 is "1," whereas, for non-error and double-bit errors, S0 is "0." Based on the generated flag signal, we can choose between the Single-Error (SE) corrector and Double-Error (DE) corrector. In this design, the SE corrector uses Hamming Single Error Correction (SEC) code and the DE uses the Double Error Correction (DEC) BCH C code

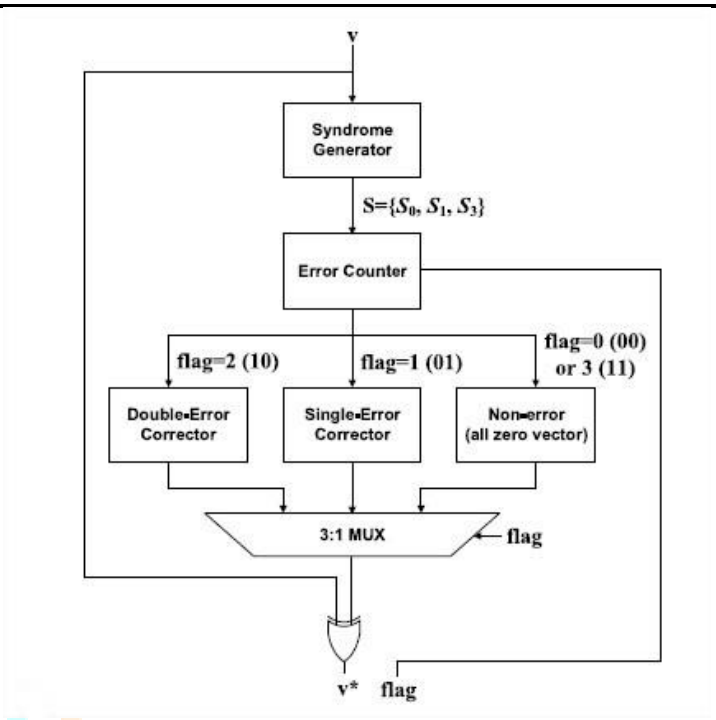


Fig. 2: Flow chart of Adaptive Error correction technique

The Fig. 2 shows the flow chart of the adaptive error correction technique, when a single-bit error occurs (flag = 01), the SE corrector, which compares each column of the $\mathbf{H_1}$ matrix with the S_1 vector, carries out a single-bit error correction. When a double-bit error occurs (flag=10), the DE corrector activates, based on the flag the multiplexer selects the corresponding error corrector and gives the result Error vector.

For the LUT-based decoder, the power and area costs of the Single Error corrector are negligible. The pair of syndrome vectors and corresponding error patterns in the LUT for the Error Detection and Correction decoder can be divided into two parts. First one is for single-bit error cases and the other is for double-bit error cases. Thus, each part can be replaced by Single Error corrector and the Double Error corrector in the LUT-based decoder, respectively.

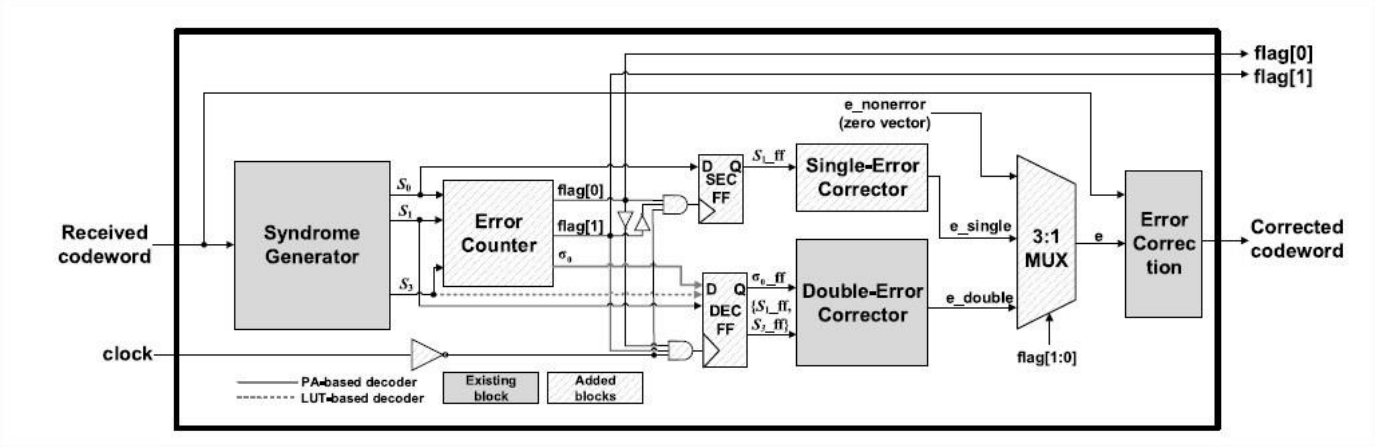


Fig. 3: Block Diagram of Double Error Corrector and Triple Error Detector.

Settled syndrome vectors should be transferred to the Single Error or Double Error corrector to prevent invalid transitions. Furthermore, the Single Error and Double Error correctors should not operate simultaneously in the proposed decoder to ensure lower power consumption. Flip-flops are used between the syndrome generator and the Single Error and Double Error correctors to satisfy these two constraints. A block diagram of the Double Error Corrector and Triple Error Detector decoder with adaptive error correction and invalid transition inhibition techniques is shown in the Fig. 3.

IV. DESIGN OF TRIPLE ERROR DETECTION AND CORRECTION (TED-TEC)

A theoretical Flow chart of the adaptive error correction technique for Triple Error Correction is shown in the Fig. 4.

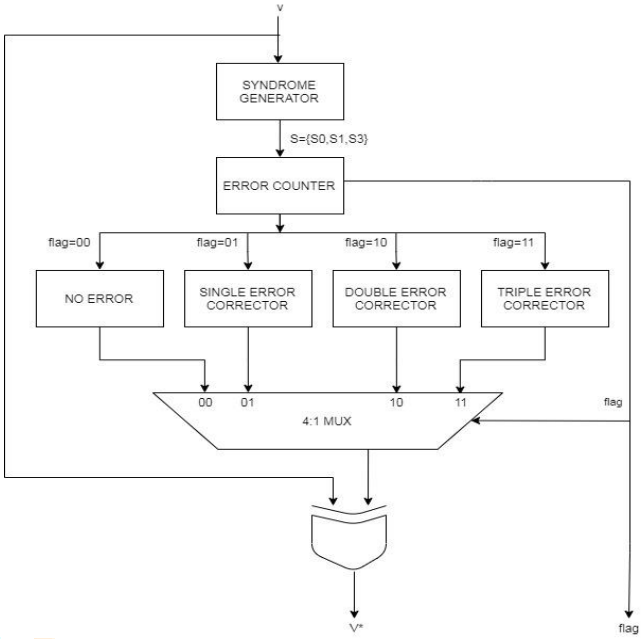


Fig. 4: Flow chart of TED-TEC.

After syndrome vectors are generated, the number of errors caused in the received codeword is classified in an error counter block, and a 2-bit flag signal that represents the number of errors is generated. Then, different error correction algorithms are applied depending on the generated flag signal to improve the decoding efficiency, and a proper error vector is added to the received codeword through the 4:1 MUX.

ALGORITHM:

Step-1: After syndrome generation we will get s0, s1, s3.
Step-2: Send s0, s1, s3 to error counter t generate error flag bits.
Step-3: By using error flag bits conditionally we select path error correction bit.
Step-4: Here if flag bits are 00 then it is sent to no error block.
Step-5: If it is 01 then it is sent to a single error correction block.
Step-6: if it is 10 it means the double error is present in information then it is sent to double error corrector.
Step-7: If it is 11 then triple bit error detected then it information send to triple error correction block.
Step-8: After error correction blocks it will select final output from mux.

To correct triple errors the Block Diagram is modified by adding extra blocks like triple error corrector flip flop and triple error correction.

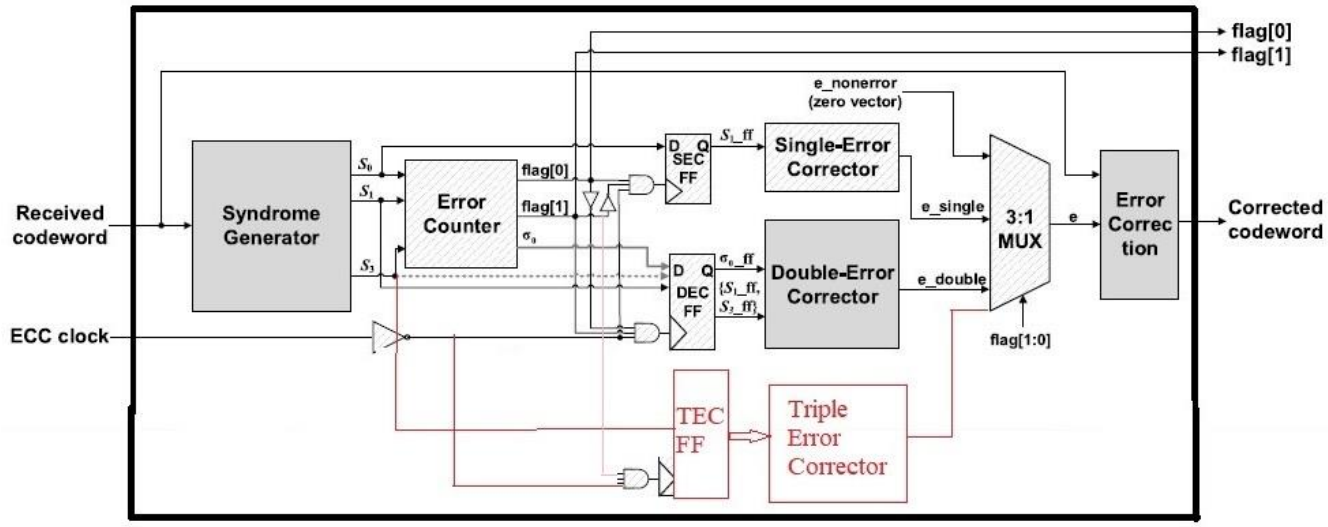


Fig. 5: Block Diagram of Triple Error Detector and Corrector.

The Block Diagram of the triple error detector and corrector is shown in the Figure 5. Positive-edge-triggered flip flops are used in this design. Flip-Flops connected to the Single Error corrector, Double Error corrector, and Triple Error corrector is called SEC-FFs, DEC-FFs, and TEC-FFs respectively for easy representation.

To make sure that both FFs transfer the settled syndrome vectors, the control signals of both FFs should be activated after the syndrome vector and flag bits become stable. To prevent the simultaneous operation of single error corrector, double error corrector, and triple error corrector flip-flops, a clock-gating technique is applied to the FF clock signal and flag bits using simple INV and the AND gates.

The single error corrector flip-flops convey the settled S_1 vector to the single error corrector only when a single-bit error occurs. Double error corrector flip-flops and triple error corrector flip-flops do not transfer the syndrome vectors to the DE or TE corrector respectively. Thus, power consumption is significantly reduced in the single-bit error case. Similarly, for double-bit and Triple-bit error cases.

V. Simulation and Synthesis Results of BCH Decoder

In this section, simulation results and synthesis of the BCH Decoder are discussed. In this work, the galois field taken as GF (2^5) then,

Block length 'n'=2^5-1=31 bits.
Number of information bits 'k'=n-mt =31-5 =26 bits (if '1' error)
=31-10 =21 bits (if '2' errors)
=31-15 =16 bits (if '3' errors).

A. Simulation of Single-bit Error Correction

The simulation of Single-bit, Double-bit, and Triple-bit error Detection and Correction are shown in the Figure 6,7 and 8 respectively.

In Single-bit error correction, the block length is '31' bits and number of information bits are '26' bits.

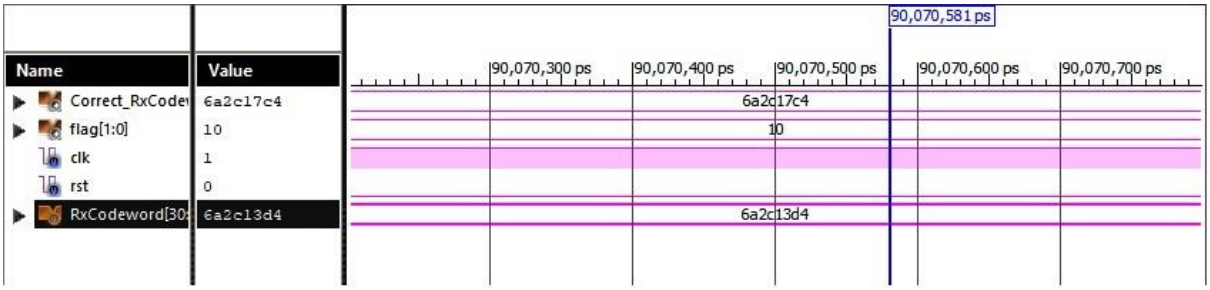
The Single-bit error occurs when one bit of the codeword stuck at bit '1' or bit '0'. The above waveforms show the simulation result of single-bit error detection and correction. In the first simulation result, the received codeword of the BCH decoder is '57e31eaf' and it is corrected into the corrected receiver codeword as '57e39eaf'. The 15th position from LSB is stuck at '0' and it is corrected to '1'.

In the second simulation result, the received codeword of the BCH decoder is '229d0b45' and it is corrected into the corrected receiver codeword as '22bd045'. The 21st position from LSB is stuck at '0' and it is corrected to '1'.

Fig. 6: Simulation waveforms of Single-bit Error Correction.

B. Simulation of Double-bit Error Correction

In Double-bit error correction, the block length is '31' bits and number of information bits are '21' bits. The Double-bit error occurs when two bits of the codeword stuck at bit '1' or bit '0'.



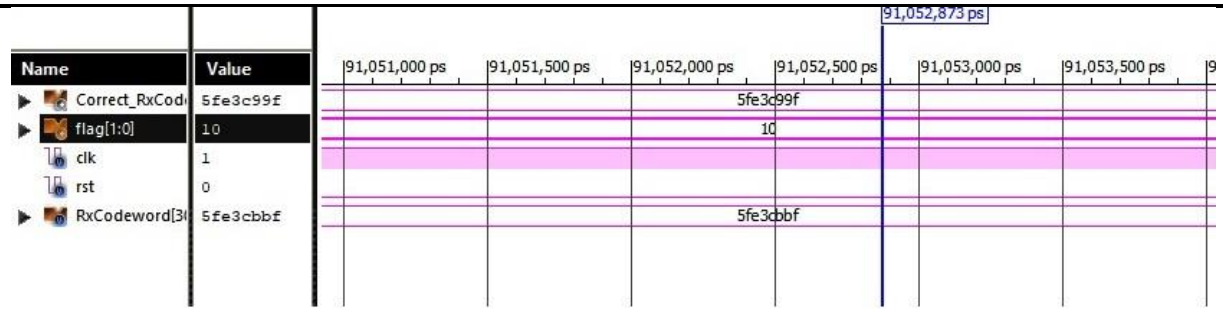


Fig. 7: Simulation waveforms of Double-bit Error Correction.

In the first simulation result, the received codeword of the BCH decoder is '6a2c13d4' and it is corrected into the corrected receiver codeword as '6a2c17c4'. The 4th and 10th bit positions from LSB is stuck at '1' and it is corrected to '0'.

In the Second simulation result, the received codeword of the BCH decoder is '5fe3cbbf' and it is corrected into the corrected receiver codeword as '5fe3c99f'. The 5th and 9th bit positions from LSB is stuck at '1' and it is corrected to '0'.

C. Simulation of Triple-bit Error Correction

In Triple-bit error correction, the block length is '31' bits and number of information bits are '16' bits. The Double-bit error occurs when three bits of the codeword stuck at bit '1' or bit '0'.

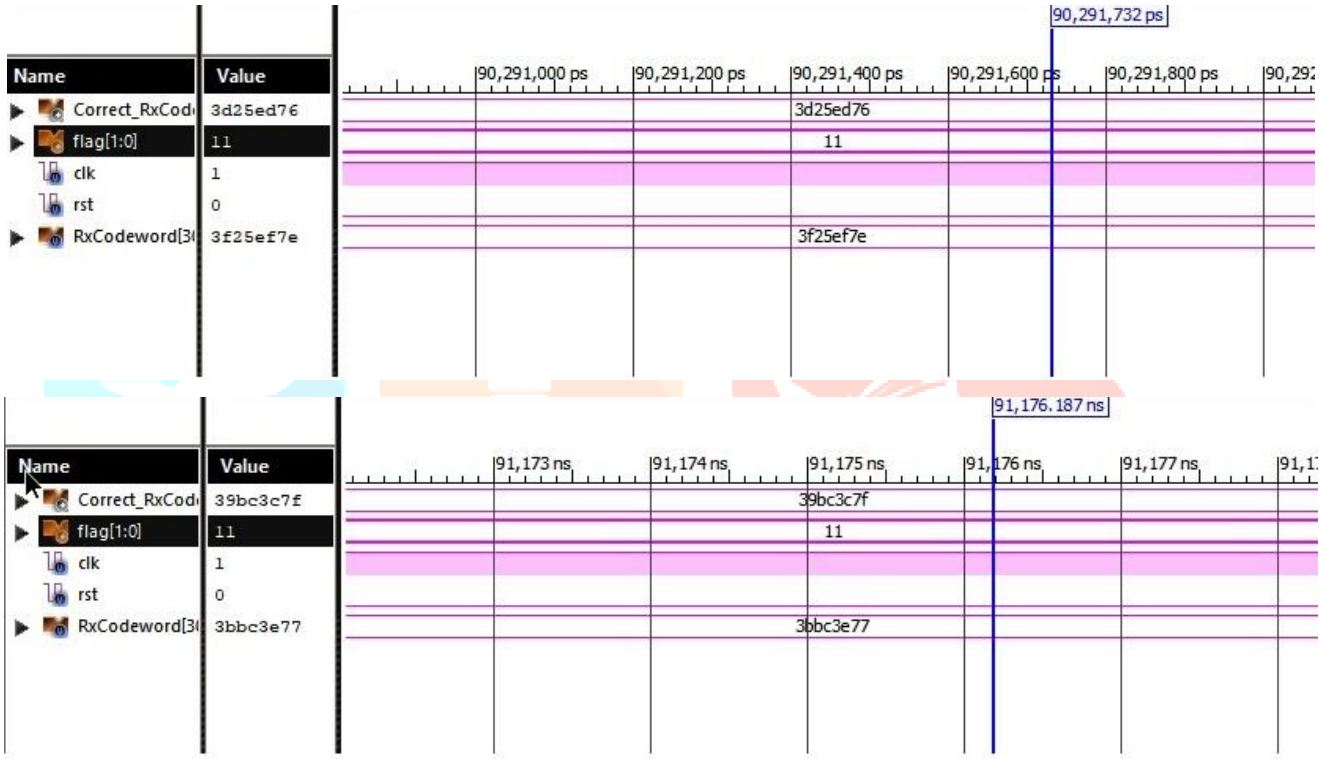


Fig. 8: Simulation waveforms of Triple-bit Error Correction.

In the first simulation result, the received codeword of the BCH decoder is '35f25ef7e' and it is corrected into the corrected receiver codeword as '3d25ed76'. The 3rd, 9th and 25th bit positions from LSB is stuck at '1' and it is corrected to '0'.

In the Second simulation result, the received codeword of the BCH decoder is '3bbc3e77' and it is corrected into the corrected receiver codeword as '39bc3c7f'. The 5th and 25th bit positions from LSB is stuck at '1' and it is corrected to '0'. The 3rd bit position from LSB is stuck at '0' and it is corrected to '1'.

D. RTL Schematic of TEC-TED BCH Decoder

In digital electronics, Register-Transfer Level (RTL) converts a digital design model into the flow of signals between hardware registers, and the logical operations performed on those signals. RTL is used in hardware description languages (HDLs) like VHDL and Verilog to create high-level representations of a circuit.

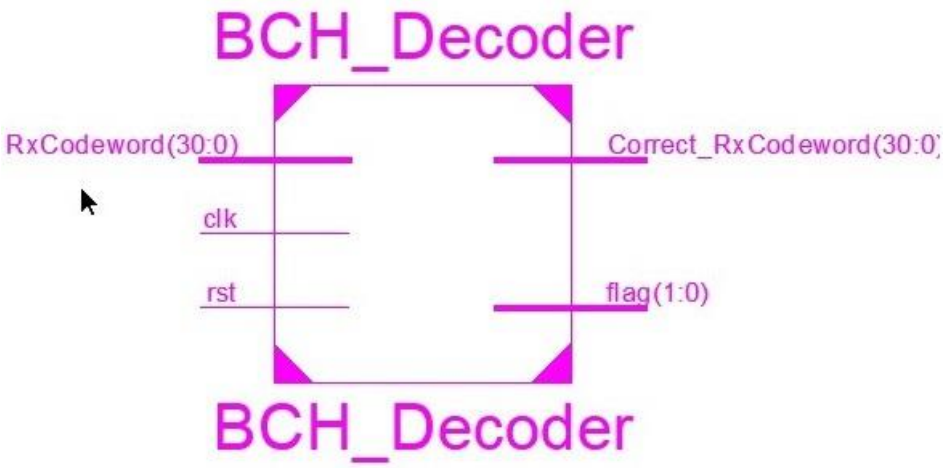
The Figure 9 shows the input and output signals used for Decoder block. The signals used in the waveform are given as

Input signals:

- 31 bits received codeword.
- Clock – 'clk'
- Reset – 'reset'

Output signals:

- 31 bits corrected codeword



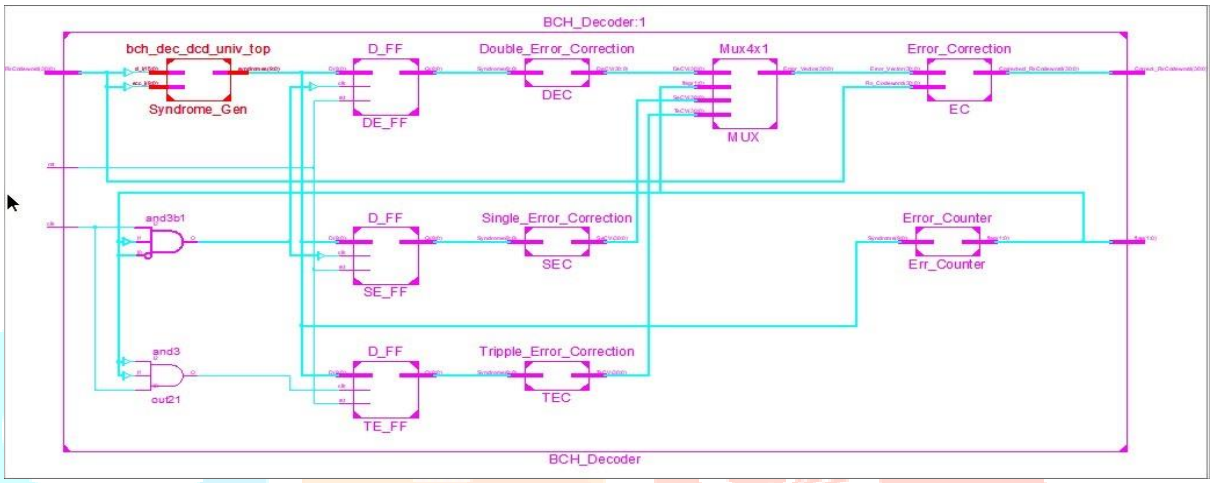


Fig. 9: RTL schematic of TED-TEC BCH Decoder

VI. CONCLUSIONS

This work covers the detailed explanation of the Error Correcting Code (ECC) and BCH Decoder design. Syndrome Generation, Error Counter, Flip-Flops, Error Corrector, Multiplexer, Exclusice-OR gate blocks are designed to form BCH Decoder. BCH Decoder is designed for single, double, triple bit Error detection and Correction. Adaptive Error Correction technique and clock gating, which are used to reduce power consumption are designed and discussed. The previous chapters discuss the design technique of decoder and the behavior of the design is described using Verilog HDL. The simulation of the code is done in Xilinx ISE.

REFERENCES

[1] Sara Choi, Hong Keun Ahn, Byung Kyu Song "A Decoder for Short BCH Codes with High Decoding Efficiency and Low Power for Emerging Memories" IEEE Transactions on Very Large-Scale Integration (VLSI) systems, Vol. 27, No. 2, February 2019.

[2] Christian Badack, Thomas Kern, Michael Gossel, "Modified DEC BCH codes for parallel correction of 3-bit errors comprising a pair of adjacent errors", IEEE 20th International On-Line Testing Symposium, August 2014.

[3] U. K. Kumar and B. S. Umashankar, "Improved Hamming Code for Error Detection and Correction", IEEE Symposium on Wireless Pervasive Computing, February 2007.

[4] J. Balakrishna, K. Suryateja and Dr. N. Alivelu Manga, "Development and Testing of ARNICC429 IP using FPGA for Navigation System", IJRECE, Volume - 6, ISSN No: 2393-9028, pp. 1387-1390, April-June 2018.

[5] Jiaqiang Li, Pedro Reviriego, Liyi Xiao, Costas Argyrides, and Jie Li "Extending 3-bit Burst Error-Correction Codes with Quadruple Adjacent Error Correction" IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 26, No. 2, February 2018.

[6] Anlei Wang, Naima Kaabouch, "FPGA Based Design of a Novel Enhanced Error Detection and Correction Technique", IEEE International Conference on Electro/ Information Technology, May 2008.

[7] Reference Textbook named "Error Control Coding: Fundamentals and Applications" by Shu Lin, Daniel J. Costello.

[8] A. R. Rishivarman, B. Parthasarathy, M. Thiagarajan "An Arithmetic over GF (2^5) to Implement in ECC", International Journal of Computer Applications, Vol. 39, No.11, February 2012.

[9] P. Reviriego, C. Argyrides, J. A. Maestro "Efficient Error Detection in Double Error Correction BCH codes for Memory Applications", ELSEVIER Research note on Microelectronics Reliability, 1528-1530, February 2012.

[10] Varinder singh, Narinder Sharma "A Review on various Error Detection and Correction methods used in Communications", American International Journal of Research in Science, Technology, Engineering and Mathematics, ISSN: 2325-3580, February 2015.

[11] M. Dug, M. Krstic, D.Jokic "Implementation and Analysis of Methods for Error Detection and Correction on FPGA", ELSEVIER Conference on International Federation of Automatic Control, July 2018.

[12] Anlei wang, Naima Kaabouch "FPGA based Design of a Novel Enhanced Error Detection and Correction Technique", IEEE Conference on Information Technology, June 2008.

[13] Jatinder Singh, Jaget Singh "A Comparative Study of Error Detection and Correction Coding Techniques", IEEE Conference on Advanced Computing and Communication Technologies, March 2012.

[14] Anil Kumar Singh "Error Detection and Correction by Hamming Code", IEEE conference on Global trends in Signal Processing and Communication, June 2017.

[15] 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, No. 5, May 2014.