# Implementation of 5 Bit Error Correcting BCH IP Core of Codelength 255 on FPGA

**Namratha N Petkar, Mr P.B Nagendra, Mrs. Prabhavathi P**

*Abstract*— **The basic requirement of the digital information and communication system is the error control coding. The errors are introduced into the digital data due to the transmission of data via a communication channel. Error correction coding is the process of correcting the errors based upon received data. Error correcting codes find application in the fields of digital data communications and memory system design.**

**In this project the BCH (Bose, Chaudhuri, and Hocquenghem) code is being implemented using a Field Programmable Gate Array (FPGA). BCH encoder and decoder are being designed and simulated using Altera Quartus and implemented in a FPGA. In this implementation, 5 bits of random error is corrected for the code of length 255.**

*Index Terms*— **BCH; FEC; Galois Field**

## I. INTRODUCTION

The demand for digital transmission and storage system has been accelerated with the rapid development and availability of VLSI technology and digital data processing[3].

In the digital systems, the fully reliable environment is expected, as the occurrence of a single error may result in shutting down of the complete system. Thus error control mechanism must be employed to detect the error and later on to correct them. The simple way of error correction is done by adding the parity bits to the original message, thus forming an encoded data[4]. This encoded data, when reaches the receiver, it is decoded to retrieve the original message.

In coding theory, the two main types of coding are Systematic code and Non-Systematic code. Any error-correcting code where the input data is inserted in the encoded output is called the Systematic code. Conversely, in Non-systematic code the input symbols are not present in the output. The advantage with the Systematic code is that the parity data can simply be appended to the input message data, and receivers need not recover the original data, if received correctly. There are two types of errors in wireless communication, namely Random errors and Burst Errors[4][3].

Manuscript received July 18, 2016

**Namratha N Petkar,** Mtech.,VLSI and Embedded Systems, BNM Institute of Technology, Bangalore, INDIA

**Mr P.B Nagendra,** Senior Engineer, D&E-TCS/Mill Com, Bharat Electronics Limited, Bangalore, INDIA

**Mrs. Prabhavathi P,** Assoc. Professor, Dept. of ECE, BNM Institute of Technology, Bangalore, INDIA

- **RANDOM ERRORS:** The transmitted errors that occur due to the presence of white Gaussian noise are referred to as Random errors.
- **BURST ERRORS:** Impulse noise is characterized by long quite intervals followed by high amplitude noise bursts. Examples of impulse noise are noise that arises due to lightning, switching transients, man-made noise etc.

Here the Random error correcting code called the BCH code is developed. This code handles randomly located errors in a data stream.

## II. FORWARD ERROR CORRECTION AND GALOIS FIELD

### A. Forward Error Correction (FEC)

The transmitter sends the information and the receiver which has the Forward error correction (FEC) technique recognizes only that portion of the input information which has no errors. As the handshaking signal between the transmitter and the receiver is not required in FEC, it can be used to broadcast the data from a single source to many destinations simultaneously. Forward Error Correction (FEC) code is able to detect a small number of random errors that occur in the received data and correct them without asking to transmit the data again. The two basic types of FEC codes are: Block codes and Convolution codes

- **BLOCK CODES:** Block codes contain (n-k) number of check bits being added to "k" bits of information to form "n" bit code. These, (n-k) check bits are generated using the "k" information bits.
- **CONVOLUTION CODES:** In convolution code, the check bits are appended to information bits continuously as the data enters the receiver. These parity check bits will help to correct errors. **Convolutional codes** are effective on bit or symbol streams of arbitrary length.

### B. Galois Field

Galois field is named after Evariste Galois, known as the finite field, where the elements in the field are finite. The data in the Galois field is represented in the binary vector format, and the mathematical operations can be performed effectively[7].

The most used operation is 'integer mod p', when p is prime.

Galois field contains a zero element called as Primitive element 'α', such that all other elements can be expressed as the power of primitive element. The existence of α is asserted by the fact that the non-zero elements of GF($p^m$) forms a cyclic group.

$p^m$ is the order of the field, and p is called the characteristic of the field.

The order of field is prime or the power of prime.
Example: GF($2^3$) = (0, 1, 2 , 2+1, $2^2$, $2^2$+1, $2^2$+2, $2^2$+2+1)
= (0,1,2,3,4,5,6,7)
GF($2^3$) has 8 elements, each polynomial of degree at most 2, evaluate at 2[7].

## III. Implementation of BCH Codes

BCH codes have got a ton of consideration as communication systems and data stored in memory systems all are in digital form, thus BCH codes are widely used. BCH codes are powerful random Error Correction Codes. BCH codes function over finite fields or Galois fields. The biggest advantage of BCH codes is the existence of efficient decoding methods due to the special algebraic structure introduced in the codes. The BCH codes are implemented as (n, k, t) codes

where, n = code length: number of bits in encoded data
k = number of bits in original message
t = error correcting capability.

For any positive integer's $t < 2m-1$ and $m \geq 3$, there exists a binary BCH code with the pursuing parametric quantities[3]:

Code length: $n = 2^m - 1$
Information bits: $k \geq n - m * t$
Minimum distance: $dmin \geq 2t + 1$
A flow chart of the operations of the BCH is shown in figure 1

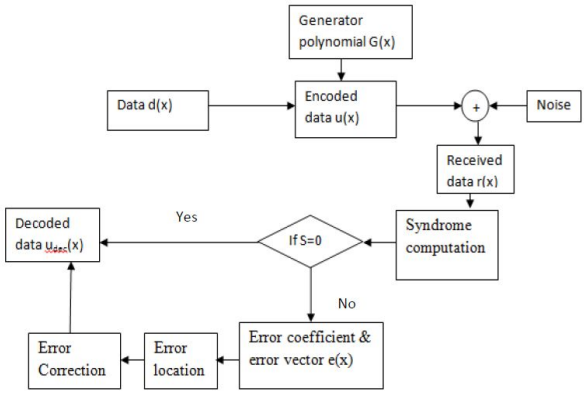


Fig: 1: The flow chart of the operation of BCH code.

The operation of BCH codes can be divided into two blocks namely, Encoder and Decoder Blocks. The Encoder block and Decoder block have many modules using which the encoding and the decoding operation is performed.
The generator polynomial of the code is indicated as its roots over the GF ($2^m$). Let α be a primitive element in GF($2^m$).
For $1 \leq i \leq t$, let $\phi_{2i-1}(x)$ be the minimum polynomial of the field element $\alpha^{2i-1}$ [5].
The generator polynomial $G(x)$ of a $t$-error-correcting primitive BCH code of length $2^m - 1$ is given by

$$G(x) = LCM \{ \phi_1(x), \phi_2(x), \ldots\ldots\ldots\ldots, \phi_{2t-1}(x) \} \quad (1)$$

### a. Encoder

The encoding function is done by generating the (n-k) parity bits using the generator polynomial and the message bits. BCH code word is encoded as:

$$c(x) = x^{n-k} * i(x) + b(x) \quad (2)$$

Where,

Codeword polynomial, $c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$,
Information polynomial $i(x) = i_0 + i_1x + \ldots + i_{k-1}x^{k-1}$,
Remainder polynomial $b(x) = b_0 + b_1x + \ldots + b_{m-1}x^{m-1}$
and $c_j, i_j, b_j \in GF(2)$.

The block diagram for the encoding circuit for (n-k) bits is as shown in the figure 2.

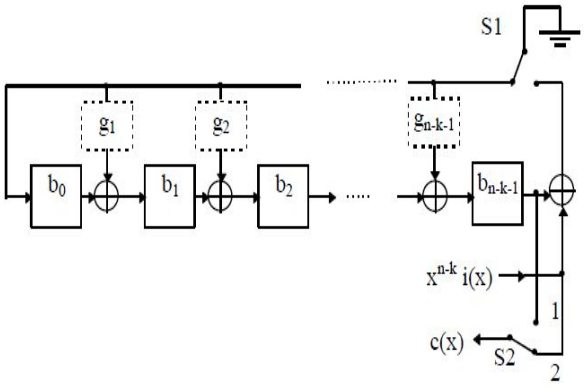


Fig: 2: The encoding circuit for (n-k) bits

When the switch 1 is closed for k bits, the message and the generator polynomial are considered. If any location of the generator polynomial is '1', then the register performs the Exclusive OR operation with the feedback. The feedback is obtained by the Exclusive OR operation of 215$^{th}$ bit of message and the 39$^{th}$ register value. If the generator polynomial is '0', then mere shift operation of the register is performed. Thus the parity bit of length 40 is obtained.
The encoded data is given as in figure 3.

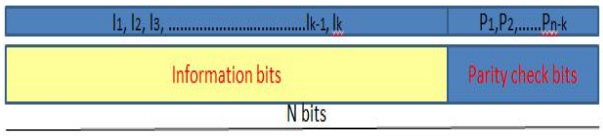


Fig: 3: The encoded data

### b. Decoder

The decoder operation can be performed in four steps, namely Syndrome computation, Error coefficients determination by Berlekamp Massey Algorithm, Error location by Chain Search Algorithm and Error correction logic[3]. The block diagram of different blocks in the decoder is as shown in figure 4.

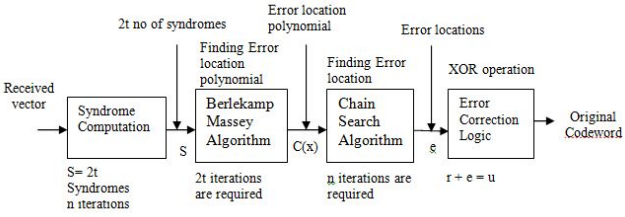


Fig: 4: The block diagram of BCH decoder

The syndrome calculator is the first module at the decoder, the design of this module is almost same for all the BCH decoder architectures. The input to the syndrome module is the received code word[3]. The received polynomial may be corrupted with error pattern e(x) as:

$$r(x) = c(x) + e(x) \quad (3)$$

where the received code word is

$r(x) = r_0 + r_1 x + r_2 x^2 + \ldots\ldots\ldots + r_{n-1} x^{n-1}$
Transmitted code word is given by:
$c(x) = c_0 + c_1 x + c_2 x^2 + \ldots\ldots\ldots + c_{n-1} x^{n-1}$
The error pattern is:
$e(x) = e_0 + e_1 x + e_2 x^2 + \ldots\ldots\ldots + e_{n-1} x^{n-1}$

For decoding the 't' error correcting BCH code, the syndrome is a 2t tuple: S=(S1,S2,…..,S2t)
Syndrome Si can be computed as:

$$Si = r(\alpha i) = r_0 + r_1 \alpha i + r_2 \alpha^2 i + \ldots\ldots + r_{(n-1)} \alpha^{(n-1)} i \qquad (4)$$

where $1 \le i \le 2t - 1$.
The $\alpha^1$, $\alpha^2$, ……. $\alpha^{(n-1)}$ are the alpha values. The alpha values are the roots of the generator polynomial obtained from the primitive polynomial. They are represented as αi where 0<i<n. Since t =5, the syndromes will be
S=(S1, S2, …… S10).

The syndrome computation is done to the received data. The syndrome values are used to find the error coefficients by using the Barlekhamp Massey algorithm.

The error vector is generated by using the error coefficients. The error location is determined by the error vector and the error correction is performed to obtain the decoded data.

Thus the decoded data is stored and compared with the encoded data. This comparison is done by computing the syndrome calculation. If the syndrome is zero then there is no error. Thus the decoded data is error free.

The Barlekhamp Massey algorithm is as given below.

Input: $S_1, S_2, \ldots, S_{10}$ (the syndromes)
Initialization:
Len=0
ELP(x)=1 (the current error locator polynomial)
PELP(x)=1 (the previous error locator polynomial)
j=1
$d_m$=1 (the previous discrepancy)
for k=1 to 10
$d = S_k + \sum_{i=1}^{Len} ELP_i S_{k-i}$ (compute discrepancy)
if d=0 (no change in polynomial)
j=j+1
else
if $2Len \ge k$ then
$ELP(x) = ELP(x) - dd_m^{-1} x^j PELP(x)$
j=j+1
else
temp(x)=ELP(x) (temporary storage)
$ELP(x) = ELP(x) - dd_m^{-1} x^j PELP(x)$
Len=k-Len
PELP(x)=temp(x)
$d_m$=d
j=1
end

Results (5)

The MATLAB code for generating primitive polynomial and the generator polynomial for the message of length 215 is written and the results are observed. The m value taken is 8. The output is seen as a Galois Field value.
The primitive polynomial for n=254 and k =214 and t=5 is generated in the MATLAB.
The generator polynomial is generated by using the primitive polynomial. Here the generator polynomial is

| | |
|---|---|
| 10100011110000001010011001101010010011 | X"4781489A93" |

The 215 bit input message for the proposed IP core is generated by concatenating the 8 bit input data 26 times and appending it by 7 bits of logic 0.
The encoder circuit generates the parity bits using the message and the generator polynomial. The parity bit is of length (n-k) i.e. (255-215) = 40 bits. The encoder output is the concatenation of the 215 bits message and 40 bits parity, thus forming 255 bit encoded data. The generator polynomial, parity bits X"390231CCA6" and the encoded data for the input "11111111" is as shown in the figure 5.

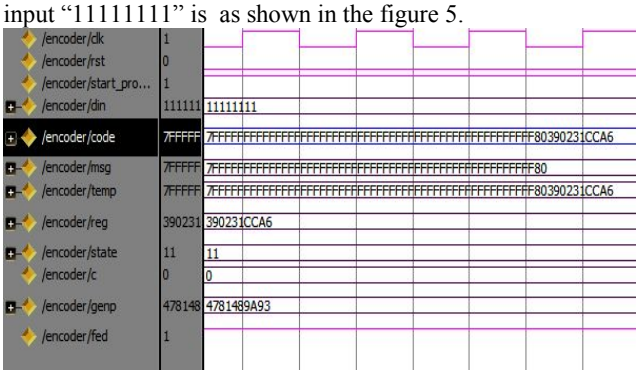

Fig 5 Encoded data

The syndrome is calculated for the encoded data with the error incorporated in it. The error coefficients are calculated using the syndrome and thus the error vector is generated using the Berlekamp Maessy Algorithm. The error coefficients calculated to the above syndrome, the error vector and the error locations are found out.

The error correction is done by performing the XOR operation to the error vector and the corrupted data. Thus the decoded data is produced as shown in figure 6. But the verification for the errorless data is done by calculation the syndrome to the decoded data. If the syndrome values are zero, then the decoded data is error free.

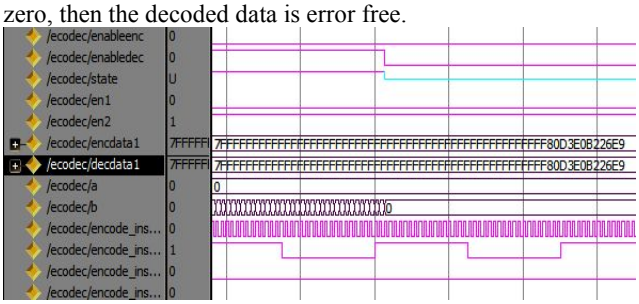

Fig 6: Decoded data

The code is being dumped on the Cyclone III FPGA, the JTAG is used to transfer the data to and from FPGA and computer. The encoded data from the FPGA is seen on the Signal Trap Logic Analyzer, the tool from Altera Quarts to view the results from FPGA. The encoded data is as shown in figure 7.

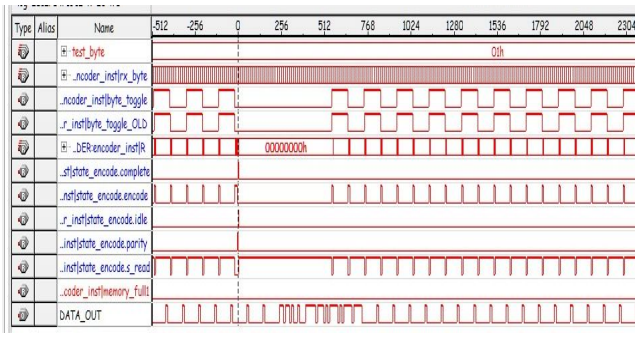


Fig 7: Encoded data seen on Signal trap Analyzer.

The decoded data, with syndrome calculation, error location and correction logic is seen on the Signal Trap Logic Analyzer as shown in figure 8.

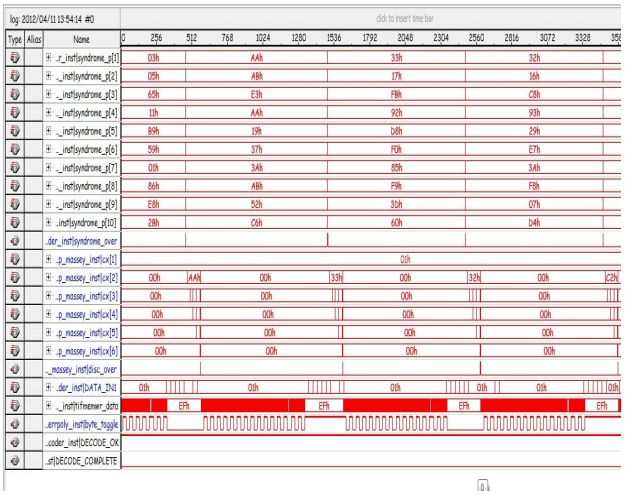


Fig 8: Decoded data seen on Signal trap Analyzer.

## CONCLUSION

The problem arising during transmission of the data through the channel is tried to reduce by reducing the number of error occurring and also enabling the rectification of the error.

BCH codes have been shown to be excellent error correcting codes among codes of short lengths. They are simple to encoder and relatively simple to decode. Due to these qualities, there is much interest in the exact capabilities of these codes.

The speed and device utilization can be improved by adopting parallel approach methods.

The efficiency can be improved by adopting codes of longer lengths. Due to this advantage, BCH codes are used in High-speed modems such as ADSL, XDSL and even in satellite and wireless communications.

## REFERENCES


1. Sahana C, V Anandi, "Error Detection Using Binary Bch (255, 215, 5) Codes", IJESRT, ISSN: 2277-9655, June, 2015.
2. Yathiraj H U, Mahasiddayya R Hiremath," Implementation of BCH code (n , k) encoder and decoder for multiple error correction control", International Journal of Computer Science and Mobile Applications, Vol.2 Issue. 5, May- 2014, pg. 45-54
3. SHU.LIN, Daniel J Costello.Jr, Error Control Coding: Fundamentals and Application.
4. Jorge Castineira Morura, Patrick Grey Farrel, Essentials of Error Control Coding, Published by John Wiley and Sons.
5. Yuan Jiang, A Practical Guide to Error Control Coding Using MATLAB , published by Artect house, Boston.
6. Volnei A Pedroni, Circuit Design With VHDL, published by MIT press, 2004.
7. Christoforus Juan Benvenuto, Galois Field in Cryptography, May 31, 2014
8. Dr.Ravi Shankar Mishra, Prof Puran Gour, Mohd Abdullah, "Design & Implementation of 8Bit Galois Encoder for on FPGA Secure Data Transmission", International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol. 1, Issue 3, pp.820-823.
9. Laurentiu Mihai Ionescu, Constantin Anton, Ion Tutanescu, Alin Mazare and Gheorhe Serban, " Hardware Implementation of BCH error Correcting Codes On FPGA", International Journal of Intelligent Computing Research (IJICR), Volume 1, Issue 3, June 2010.
10. R Elumalai, A Ramachandran, J V Alamelu, Vibha B Raj, " Encoder and Decoder for (15,11,3) and (69,39,4) Binary BCH Code With Multiple Error Correction", International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering Vol. 3, Issue 3, March 2014.
11. Priya Mathew, Lismi Augustine, Sabarinath G, Tomson Devis, " Hardware Implementation Of (63,51) BCH Encoder and Decoder for WBAN using LFSR and BMA", International Journal on Information Theory (IJIT), Vol.3, No.3, July 2014.
12. Habti Idrissi Anas , Gouri Rachid, Ahmed Lichioui, Hlou Laamari, " Conception of a new Syndrome Block for BCH codes with hardware Implementation on FPGA Card"., Int. Journal of Engineering Research and Applicationswww.ijera.com ISSN : 2248-9622, Vol. 5, Issue 5, ( Part -2) May 2015, pp.80-85 .
13. Sirath. A, Sharqa Ekram, Sinky Kumari , "Error Control Circuit And Its Implementation On Radio Link Enhancement". Vol. 2 Issue 4, April – 2013 (IJERT).
14. Samir Jasam Mohammed, Hayder Fadhil Abdulsada , "FPGA Implementation of 3 bits BCH Error Correcting Codes", (IJERT). Volume 71– No.7, May 2013.
15. Rohith S, Pavithra S , "Fpga Implementation Of (15,7) Bch Encoder And Decoder For Text Message" . IJRET eISSN: 2319-1163 | pISSN: 2321-7308
16. Shannon, C. E., "A mathematical theory of communication," Bell Syst. Tech. J., vol. 27,pp. 379–423, 623–656, July and October 1948.