IJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 5 Issue 1 January, 2017

IMPLEMENTATION OF BCH LFSR ENCODER DECODER

K. Manoranjan[1]M.Chennaiah[2]; K.PrasadBabu[3]; S.AhmedBasha [4]K.Sudhakar [5]

1M.Tech Scholar, Department of ECE, SJCET, Yerrakota, JNTUA University, AP, India

2Associate Professor & Head, Department of ECE, SJCET, Yerrakota, JNTUA University, AP, India

3-4Assistant Professor, Department of ECE, SJCET, yerrakota, JNTUA University, AP, India

manoranjan9949@gmail.com [1];chennaiah.m@gmail.com[2],kprasadbabuece433@gmail.com[3]; ahmedbasha.syed@gmail.com[4],sudhakar_403@yahoo.co.in [5]

ABSTRACT

BCH codes is an acronym for Bose, Ray – Chaudhuri, Hocquenghem, invented in 1960s and today they are used as a baseline for many recent Error Correcting Codes. BCH codes are powerful class of multiple error correction codes with well defined mathematical properties. BCH code is used to correct multiple random error patterns. The mathematical properties within which BCH codes are defined are the Galois Field or Finite Field Theory. The main focus of this project is to design encoder and decoder architecture for BCH codes. These types of codes are used in communications networks to detect and correct errors. The design of an encoder is based on Linear Feed Back Shift Register used for polynomial division and the decoder design is based on the algorithm to correct the errors occurred during transmission. The combination of BCH codes and LDPC codes are used for error correction for satellite communication standards. The proposed decoder supports the multi-byte parallel operations to enhance its throughput. In addition, it employs a LFSR-based parallel syndrome generator for compact hardware design

Keywords: Barrel Decoder, Encoder, Error Correction; BCH Code,LFSR.

1. INTRODUCTION

Data corruption during the transmission and reception of data because of noisy channel medium is the most common problem faced in digital communication system. Thus, it is hard to get the reliable communication. Thus, to get the error free communication we need Error correction code. BCH codes is an acronym for Bose, Ray – Chaudhuri, Hocquenghem, invented in 1960s and today they are used as a baseline for many recent Error Correcting Codes. BCH codes are powerful class of multiple error correction codes with well defined mathematical properties. BCH code is used to correct multiple random error patterns. The mathematical properties within which BCH codes are defined are the Galois Field or Finite Field Theory. The main focus of this project is to design encoder and decoder architecture for BCH codes. The design of an encoder is based on Liner Feed Back Shift Register used for polynomial division and the decoder design is based on ibm algorithm to correct the errors occurred during transmission. Also this project report contains comparison of BCH codes with other Error Correcting codes and gives the detailed explanation of salient feature of BCH codes. The combination of BCH codes and LDPC codes are used for error correction for satellite communication standards.

The BCH codes architecture is described using hardware description language called Verilog and synthesized using Xilinx 9.1 ISE. The performance of the whole model is check in terms of simulation using Xilinx simulator

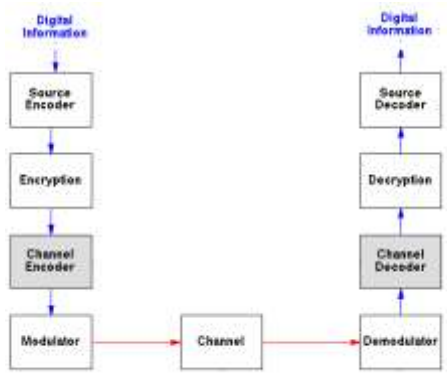


Figure 1: Block Diagram of basic communication system

Steps:

[1] In the 1st step the message bits are given as input to the Source encoder_LFSR. This is the module where the density of message bits is increased by compression. This module use BCH codes or other Error correction code to encode the compressed data by adding the parity bits. The output of the source encoder_LFSR is a longer information bits, called "codeword".

[2] This codeword is given as input to the Encrypted module. This module increases the security of the information bits.

[3] The 3rd module is channel encoder_LFSR, which increases the reliability and the hardware complexity of the Codeword.

[4] The output of the channel encoder_LFSR is fed to modulator, which transform the codeword into a signal waveform. This modulated signal is now ready to transmit through a Channel.

[5] The channel could be a physical medium or a wireless medium. The possibility that the codeword get corrupted in the channel is high because the channel is subjected to various noises.

[6] The corrupted signal is received at the Demodulator module at the receiver section. In this module the signal, get de-modulated to get the corrupted codeword.

[7] The original message bits are retrieved from the corrupted codeword using the decoding algorithm of Error correction code (ECC), which was initially used to encode the data at the Encoder_LFSR. Thus, the original information bits are retrieved at the output of the Source decoder [5] [6].

2. IMPLEMENTAION

In recent years there has been an increasing demand for digital transmission and storage systems. This demand has been accelerated by the rapid development and availability of VLSI technology and digital processing. It is frequently the case that a digital system must be fully reliable, as a single error may shutdown the whole system, or cause unacceptable corruption of data, e.g. in a bank account . In situations such as this error control must be employed so that an error may be detected and afterwards corrected. The simplest way of detecting a single error is a parity checksum [2], which can be implemented using only exclusive-or gates. But in some applications this method is insufficient and a more sophisticated error control strategy must be implemented. If a transmission system can transfer data in both directions, an error control strategy may be determined by detecting an error and then, if an error has occurred, retransmitting the corrupteddata . These systems are called automatic repeat request (ARQ). If transmission takes place in only one direction, e.g. information recorded on a compact disk, the only way to accomplish error control is with forward error correction (FEC) . In FEC systems some redundant data is concatenated with the information data in order to allow for the detection and correction of the corrupted data without having to retransmit it. One of the most important classes of FEC codes is linear block codes [2,29,43]. In block codes, data is transmitted and corrected within one block (codeword). That is, the data preceding or following a transmitted codeword does not influence the current codeword. Linear block codes are described by the integer n, the total number of symbols in the associated codeword. Block codes are also described by the number k of information symbols within a codeword, and the number of redundant (check) symbols n-k.In error control, it is crucial to understand the sources of errors. Each transmitted bit has probability $p \geq 0$ of being received incorrectly. On memoryless channels every transmitted symbol may be considered independently, so only random errors occur. Unfortunately, most channels have memory and usually several successive symbols are corrupted. These kinds of errors are called burst errors [29]. Burst errors can be most efficiently corrected through use of burst error correcting codes, e.g. Reed Solomon (RS) codes . Because the structure of burst error correcting codes is usually complicated, multiple random error correcting codes are often employed. In order to improve burst error correction, the transmitted codewords are also rearranged by interleaving. The resulting code is called an interleaved code. In this way the burst errors scatter into several codewords and look like random errors. Other operations on block codes are also available to improve the error correcting ability or to adapt a code to a specified requirement. For example codes may be shortened, extended, concatenated or interleaved [2,5]. The simplest block codes are Hamming codes. They are capable of correcting only one random error and therefore are not practically useful, unless a simple error control circuit is required. More sophisticated error correcting codes are the Bose, Chaudhuri and Hocquenghem (BCH) codes that are a generalisation of the Hamming codes for multiple-error correction. In this thesis the subclass of binary, random error correcting BCH codes is considered, hereafter called BCH codes. BCH codes operate over finite fields or Galois fields. The mathematical background concerning finite fields is well specified and in recent years the hardware implementation of finite fields has been extensively studied [4,5,6,7,8,9,10,13,]. Furthermore, any BCH code can be defined by only two fundamental parameters and these parameters can be selected by the designer. These parameters are crucial to the design and the question arises if it is possible to develop a tool that will automatically generate any BCH codec description, just by providing the code size n and the number of errors to be corrected t. This design automation would considerably reduce BCH codec design cost and time and increase the ease with which BCH codecs with different design parameters are generated. This is an important motivation since the architectures of BCH codecs with different parameters can vary remarkablyBCH codesThree different decoding strategies are presented according to the error correcting capability of the code. Generally decoding is broken down into three processes, syndromes calculation, Berlekamp-Massey algorithm (BMA) and Chien search.

The format of the codeword is as follows [4]:

c(x) = xn-k * i(x) + b(x) (1)

Where, codeword c(x) = c0 + c1x +...+ cn-1xn-1

information bits i(x)= i0 + i1x +...+ ik-1xk-1

remainder b(x)= b0 + b1x +...+ bm-1xm-1

also, ci, ji, bi are the subsets of Galois field

Basically, encoder_LFSRmodule consists of three modules.

[1] 5 bit Parallel to Serial Shift Register

[2] Encoder_LFSR module - Linear feedback shift register

[3] Serial to Parallel Shift Register

The Encoder_LFSR module is designed with respect to the generated polynomial given Generator polynomial is divided by the incoming 5-bits message bits. The division is carried out using linear feedback shift register, the remainders of this division is added with the original message bits to form a "codeword".

The Decoder design is consists of 5 blocks. These blocks are:

1. Parallel to Serial Shifter
2. Syndrome Block
3. Inversion-less Berlekam Massey Block
4. Chain Search Block
5. Error Correction Block.
6. Syndrome Block

The output of the parallel to series shifter is given as input to the Syndrome block. As a first step of syndrome block operation the input 1bit is converted to 4-bits. These 4-bits are given as input to the six parallel syndromes blocks. These syndrome blocks generate the syndrome S1_IN, S2_IN, S3_IN, S4_IN, S5_IN and S6_IN. The architecture of the syndrome blocks is given as per the equation (3.7). In the first clock pulse these 4-bits are added with α(i) and this result is stored in the register D. At the next clock pulse the result stored in register D is multiplied by α(i) and the resultant bits of this multiplication is added with new incoming expanded 4-bits. The register D is now updated with the new resultant bits. This cyclic steps repeat in all six parallel syndromes for 15 clock cycles, until the last bit of the corrupted codeword is enters to the syndrome block. The output syndromes S(i) is generated at the end of 16th clock cycle. The hld signal for the syndrome block is set to high at the 18th clock pulse to make sure that the output of the syndrome block is not changed with every clock cycle.

FIGURES/CAPTIONS

.

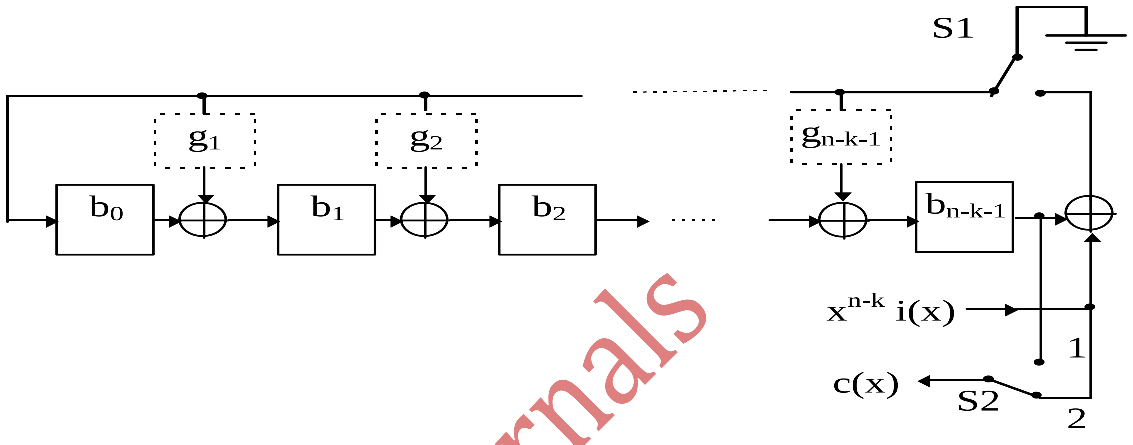


Fig 2: Encoding circuit for BCH

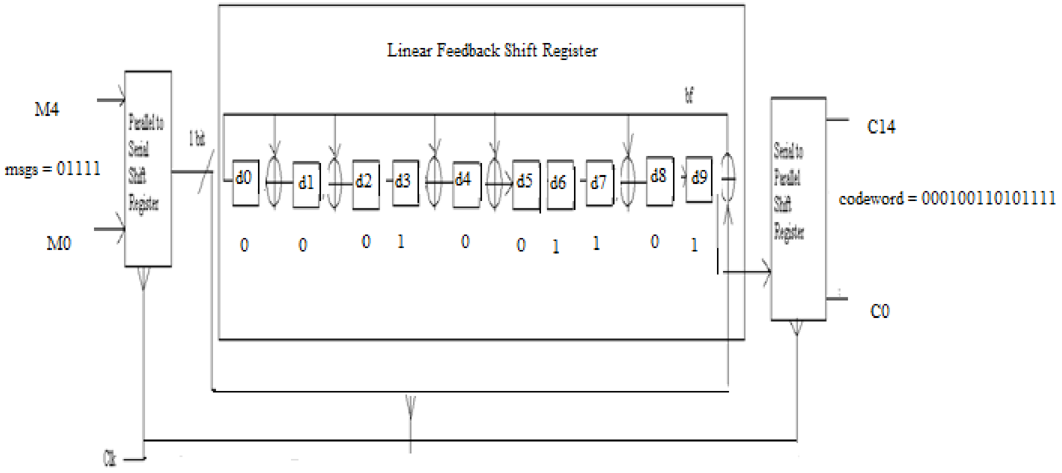

Linear Feedback Shift Register
M4
msgs = 01111
M0
Parallel to Serial Shift Register
1 bit
d0
d1
d2
d3
d4
d5
d6
d7
d8
d9
0 0 0 1 0 0 1 1 0 1
Serial to Parallel Shift Register
C14
codeword = 000100110101111
C0
Clk

Fig 3: Encoder_LFSR

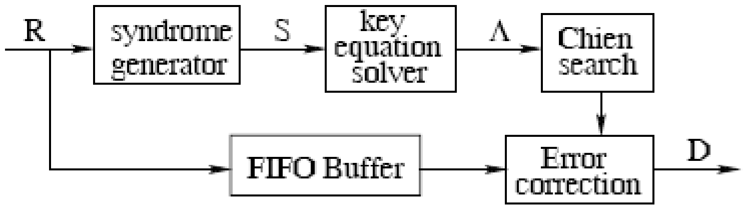


Fig 4: Decoder

R = Received data, S = Generated Syndromes, Λ = Error locator polynomial

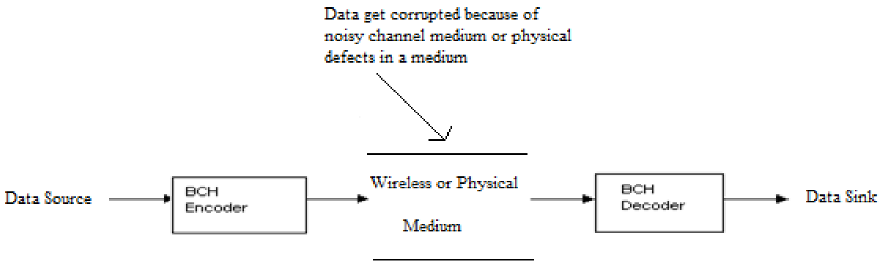

Data get corrupted because of noisy channel medium or physical defects in a medium
Data Source
BCH Encoder
Wireless or Physical Medium
BCH Decoder
Data Sink

Fig 5: BCH Encoder_LFSR and Decoder

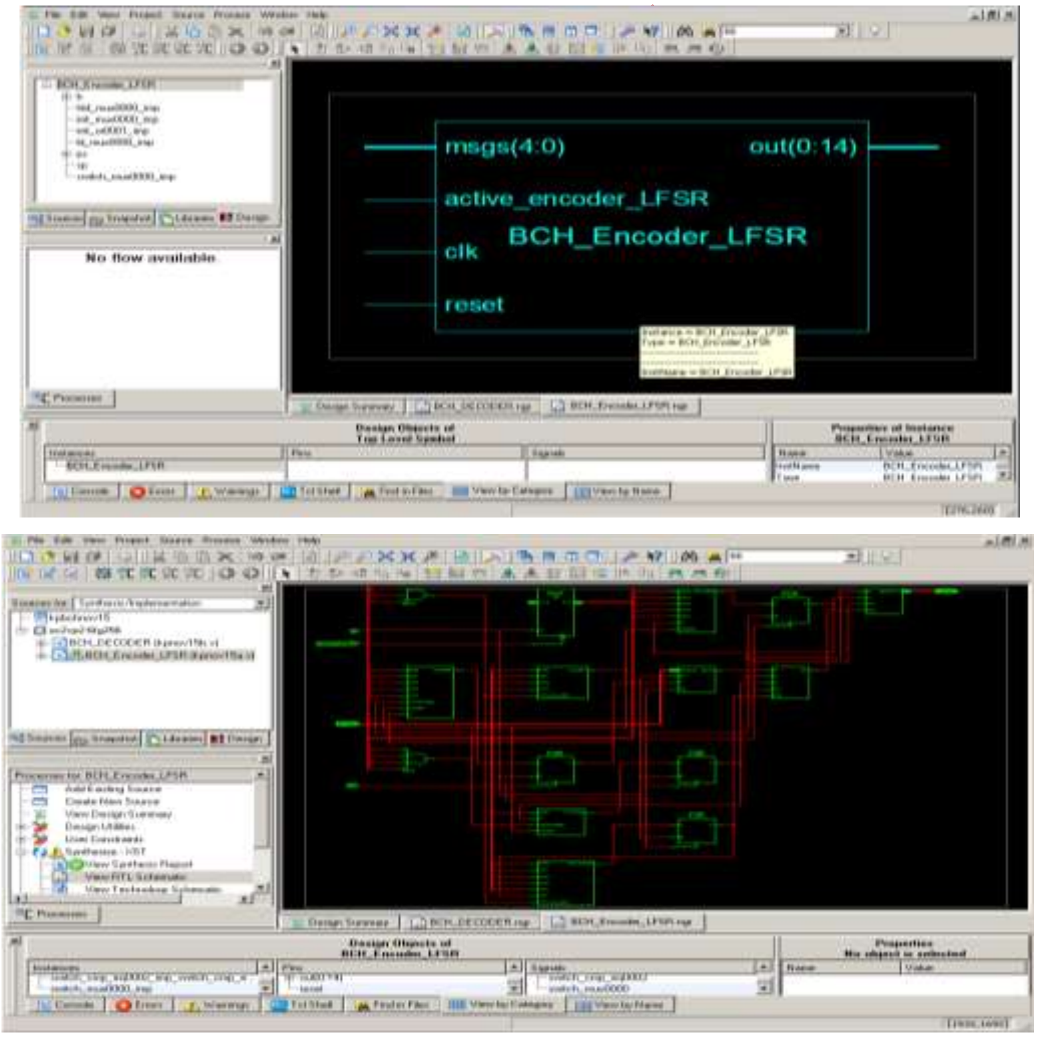
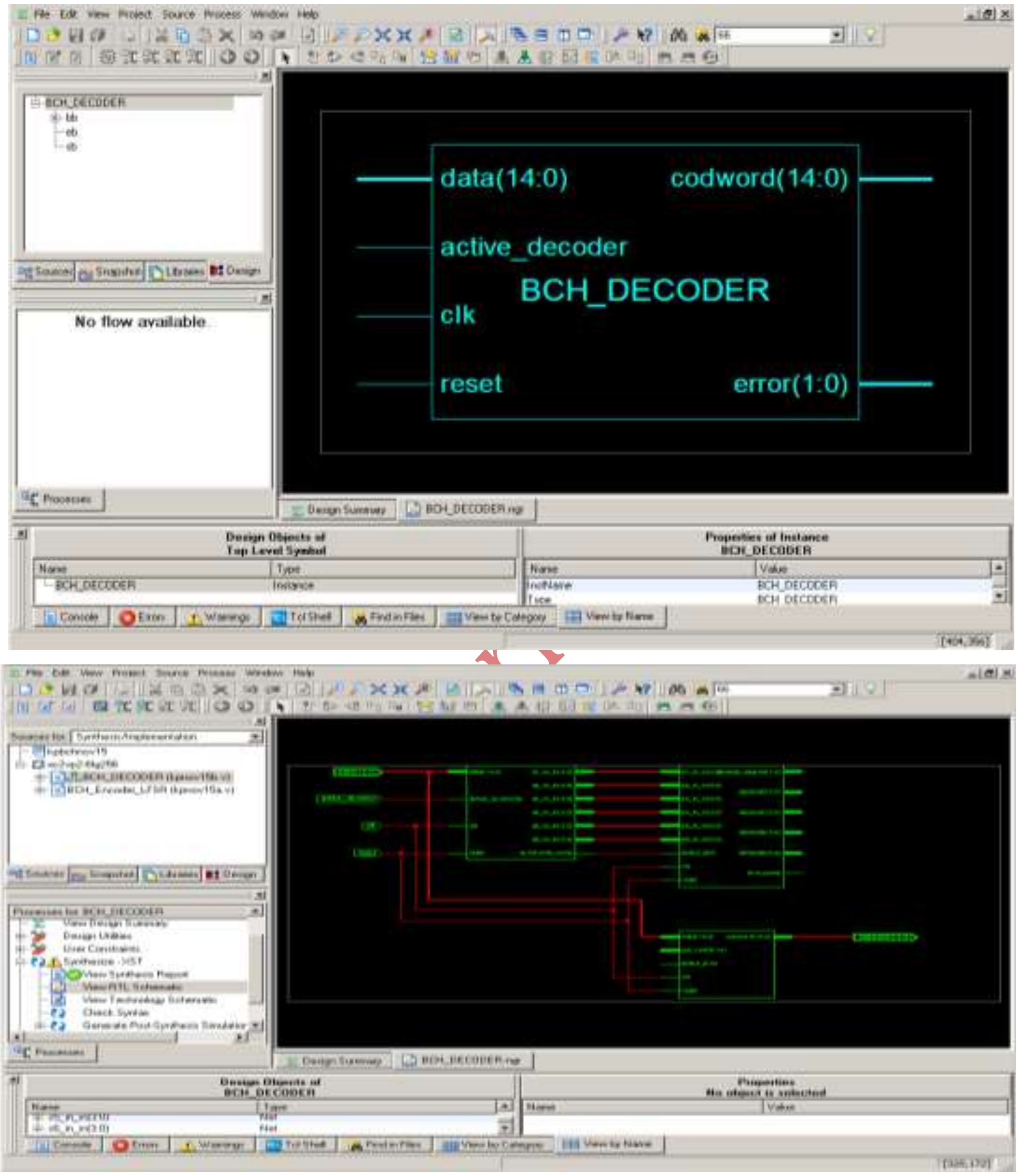


Fig 6: RTL SCHEMATIC of BCHEncoder_LFSR



Design Objects of Top Level Symbol

Name	Type
BCH_DECODER	Instance

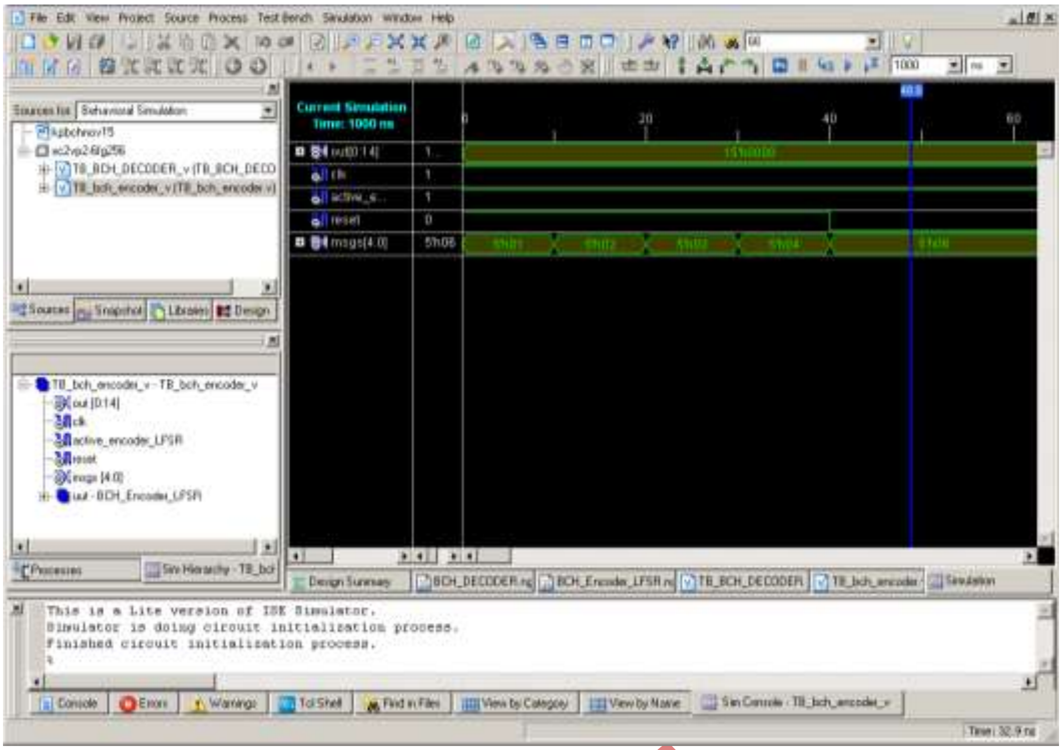
Properties of Instance BCH_DECODER

Name	Value
InstName	BCH_DECODER
Type	BCH_DECODER

Design Objects of BCH_DECODER

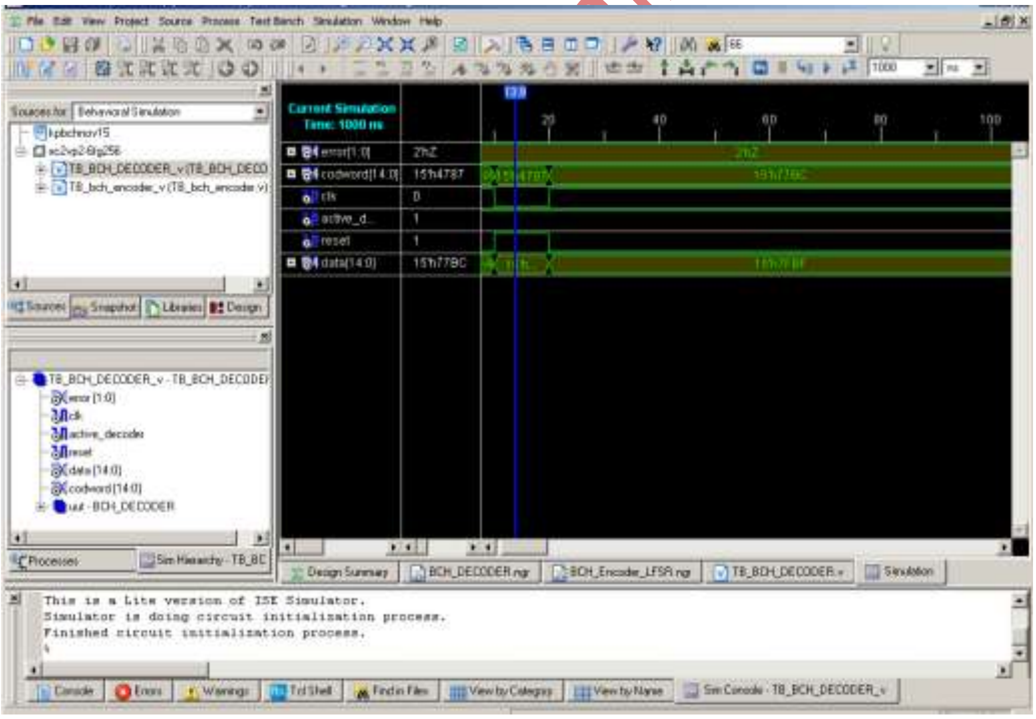
Properties No object is selected

Fig 7: RTL SCHEMATIC of BCH Decoder



```
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
%
```

Fig 8:Test-Bench wave form of BCH Encoder



```
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
%
```

Fig 9: Test-Bench wave form of BCH Decoder

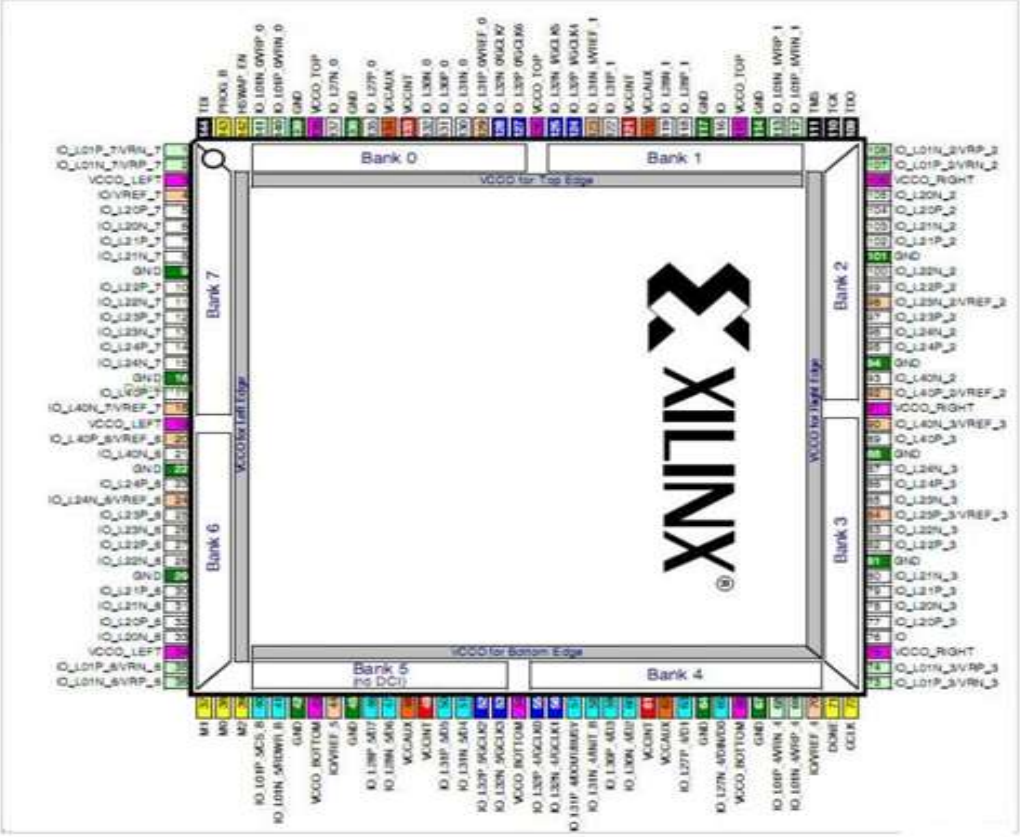


Fig 10: Pin diagram of Xilinx devicexc2vp2-6-fg256

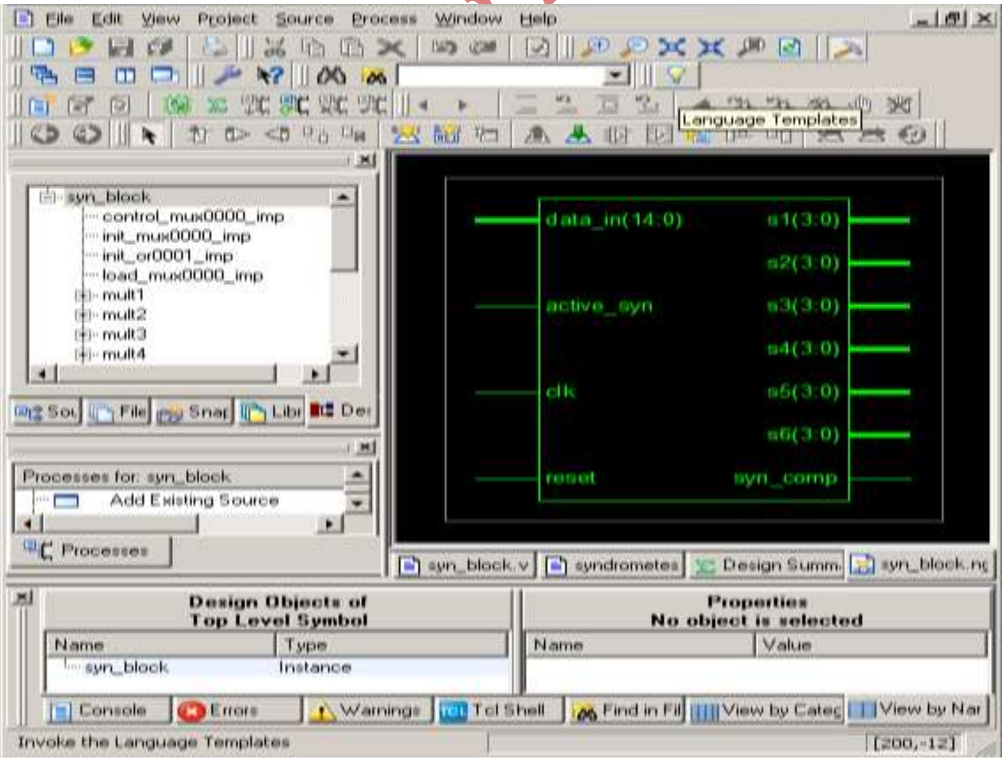
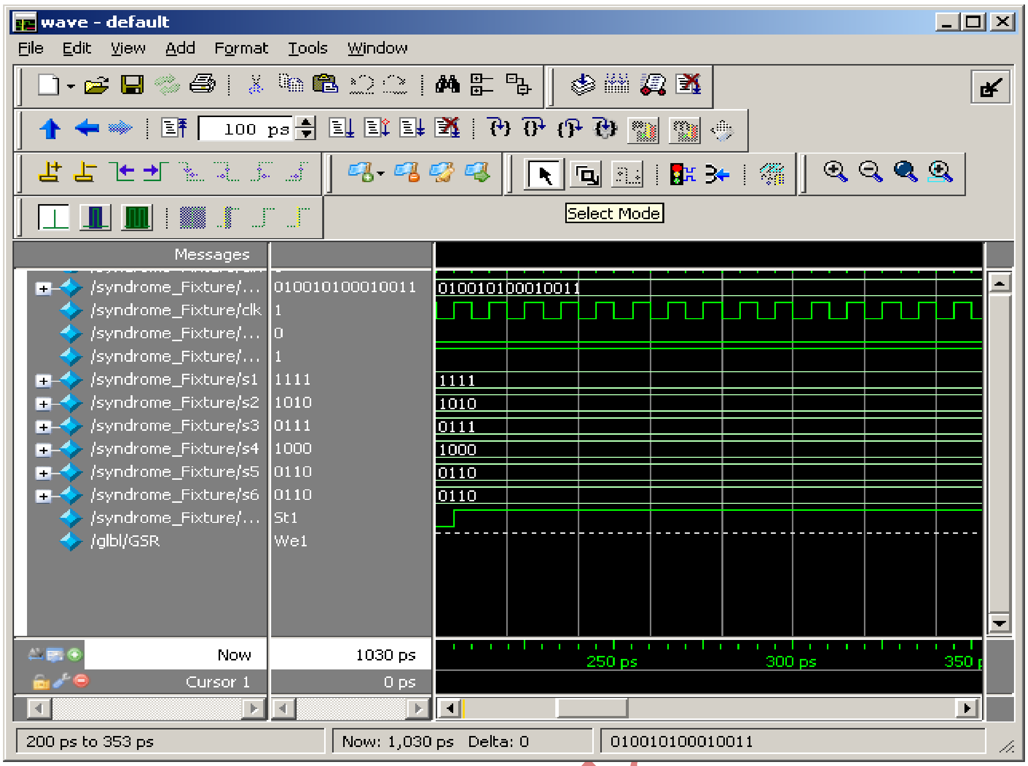


Fig 11 Syndrome block



```
/syndrome_Fixture/...   010010100010011   010010100010011
/syndrome_Fixture/clk   1
/syndrome_Fixture/...   0
/syndrome_Fixture/...   1
/syndrome_Fixture/s1    1111              1111
/syndrome_Fixture/s2    1010              1010
/syndrome_Fixture/s3    0111              0111
/syndrome_Fixture/s4    1000              1000
/syndrome_Fixture/s5    0110              0110
/syndrome_Fixture/s6    0110              0110
/syndrome_Fixture/...   St1
/glbl/GSR               We1

Now        1030 ps
Cursor 1   0 ps

200 ps to 353 ps   Now: 1,030 ps   Delta: 0   010010100010011
```

Fig 12 Test Bench wave form of Syndrome block

3. CONCLUSION

From the summary table it can be seen that the there are totally 26,624 flip flops are available in the FPGA board, out of which only 39 flip flops are being used. So, the utilization percentage of the flip flops is only 1%. Also, the available LUT are 26,624 out of which only 32 are being used, hence the utilization percentage of the LUT is also almost equal to 1%. There are 221 IOBs are available in FPGA out of which only 23 IOBs are used for encoder_LFSR design. So, the utilization percentage is 10%. Thus, it could be seen that the percentage of the various components used for the encoder_LFSR design are very less as compared to the available quantity. So, it could be concluded that the encoder_LFSR design of this project is optimized and also has good performance.For decoder, totally 1,320 flip flops are available in the FPGA board, out of which only 213 flip flops are being used. So, the utilization percentage of the flip flops is only 11%. Also, the available LUT are 1,320 out of which only 560 are being used, hence the utilization percentage of the LUT is also almost equal to 23%. There are 66 IOBs are available in FPGA out of which only 35 IOBs are used for decoder design. So, the utilization percentage is 53%. Thus, it could be seen that the percentage of the various components used for the BCH_Decoder design are very less as compared to the available quantity. So, it could be concluded that the BCH_Decoder design of this project is optimized and also has good performance

4. REFERENCES

[1]. R.C. Bose, D.K. Ray-Chaudhuri, “On a class of error correcting binary group codes”, Inf. Cntrl, 3, pp. 68-79, March 1960.

[2] H.O. Burton, “Inversionless decoding of binary BCH code”, IEEE Trans., 1971, IT- 17, (4), pp. 464-466.

[3] C. E. Shannon, ``A mathematical theory of communication," Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[4] Ernest Jamro, “The Design of a VHDL based synthesis tool for BCH codes”, The university of Huddersfiel, September 1997.

[5] W.W. Peterson, E.J. Weldon, “Error correcting codes”, MIT Press, Cambridge, MA, 1972.

[6] W.W. Peterson, “Encoding and error-correction procedures for the Bose-Chaudhuri Codes”, IRE

Trans. Inf. Theory, IT-6, pp. 459-470, September 1960.

[7] Joel Sylvester "Reed Solomon Codes", Elekrobit., January 2001.

[8] G. D. Farney, lr., "The Viterbi algorithm", *Proc. IEEE, vol. 61, pp. 268-278*, Mar. 1973.

[9] Softjin technology, "Data sheet for BCH encoder_LFSR core', India, Dec 2000, www.softjin.com.

[10] Hanho Lee, "An area-efficient Euclidean Algorithm block for Reed-Solomon Decoder", *Dept. of ECE, University of Conecticut, Srorrs, CT 0* 6269, USA.

[11] Dilip V. Sarwate, Naresh R. Shanbhag, "High-Speed Architectures for Reed-Solomon Decoders," *IEEE Trans. On VLSI Systems, vol.9 No.5, Oct. 2001.*

[12] Yanni Chen, Keshab K. Parhi, "Area Efficient Parallel Decoder Architecture for Long BCH Codes", *Dept. of ECE, University of Minneapolis, MN 55455 USA*

[13] Clifford Kraft, "Closed Solution of Berlekamp's Algorithm for Fast Decoding of BCH Codes", *IEEE Transactions on Communications, Vol. 39, No. 12, December1991.*