Advance in Electronic and Electric Engineering.
ISSN 2231-1297, Volume 4, Number 3 (2014), pp. 273-278

http://www.ripublication.com/aeee.htm

FPGA Implementation of Cyclic Code Encoder and Decoder

Gaurav Chawla and Vishal Chaudhary

ECE Department, Kurukshetra University, Seth Jai Prakash Mukand Lal Institute of Engineering and Technology, Raduar, Yamunanagar, Haryana, INDIA.

Abstract

This paper presents Cyclic code Encoder and Decoder with its soft core design as well hardware implementation on FPGA. The design includes both encoder and decoder part that can be used in a communication system for encoding a message at the transmitter and decoding it, detecting error and correcting it on the receiver part. The source code for Encoder and Decoder has been formulated in VHDL (VHSIC Hardware Description Language) and simulated and synthesized using Xilinx ISE14.3 The hardware implementation has been done on Xilinx Spartan6 LX45 FPGA.

Keywords: FPGA; Cyclic Code; VHDL.

1. Introduction

The essence of a communication system depends on how it deals with the noise that may interfere with the data to be transmitted. Noise generally is encountered in the channel phase of a communication system which could be transmission lines, optical fibers, space, air etc. The coding theory is an important tool for encoding a given message, decoding and correcting the received message. Cyclic codes are one such tool that can be used both for encoding and decoding of the information represented in the form of binary numbers.

Cyclic Codes were first studied in 1957 by Prange. After that they became an important part of the coding theory. Cyclic codes are easy to study and implement because: 1) Encoding and syndrome computation can be done easily using shift registers with feedback connections. 2) Due to the inherent algebraic structure, they are easy to decode.

This Paper is organized as follows: Section II: Encoding and Decoding of Cyclic Codes; Section III: Software implementation of Cyclic Code with simulation results; Section IV: Hardware implementation of Cyclic Code; Section V: Conclusion.

2. Encoding and Decoding of Cyclic Codes

Cyclic codes are an important class of linear block codes in which the cyclic shifting of the message bits results in another code vector, hence the name cyclic code. In other words a cyclic shift in a code word in C results in another code word in C. For Example C = {(110), (101), (011)} is cyclic, while C = {(000), (100), (111)} is not cyclic. A cyclic code is defined in terms of a generator polynomial g(X) of degree less than n-k , where k is the length of input message and n is the length of the encoded message.

Cyclic codes are studied usually in polynomial form since it is easy to represent the code vectors as the coefficients of the polynomial. For example the message 110001 is represented as $1+ X + X^5$. Cyclic Codes are used both for encoding and decoding of the message bits.

In an encoding process the message signal is divided by a certain sequence called generator number and corresponding to that the remainder bits form the parity check sequence which are concatenated to either front or back of the message signal to encode it. The process of encoding can be implemented using shift registers as shown in the Fig. 1, where $g_0,g_1,..g_{n-k-1}$ is for generator bits, $b_0,b_1,..b_{n-k-1}$ is for remainder bits and u(X) is for message bits.

The decoding process is a more complex process and is followed by error detection and correction. In error detection the received vector is divided by the generator sequence and the computed remainder forms the syndrome.

If the computed syndrome is identical to zero, then the received vector is error free, else the process advances to error correction. In error correction the computed syndrome is compared with the error pattern and accordingly the erroneous bit is XOR-ed with the error bit and the corrected received vector is obtained.

The syndrome computation can be implemented using shift registers as shown in Fig. 2, where $s_0,s_1,….s_{n-k-1}$ is for syndrome.

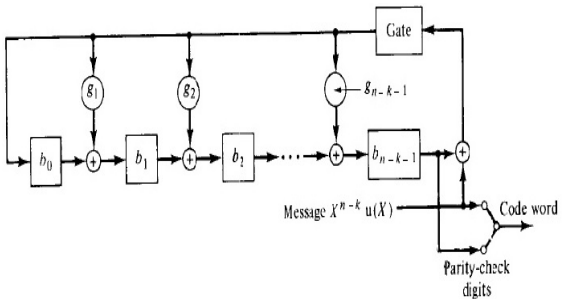

Fig. 1: Encoding circuit for (n,k) Cyclic Code (Image Courtsey: Shu Lin, 1983)

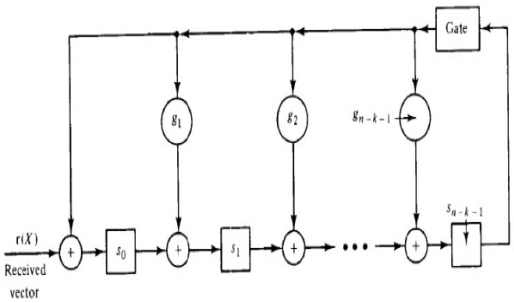

Fig. 2: Syndrome circuit for (n,k) Cyclic Code (Image Courtsey: Shu Lin, 1983)

3. Software Implementation of Cyclic Code

In the software implementation we simulated many examples, the one shown in this paper is (15,10) Cyclic Code. Here we input the message bits 1010010001 and the generator sequence 10101, the parity bits obtained are 01010 which are concatenated at the end of the message. So the encoded message is 101001000101010. The simulation of Encoder is shown in Fig. 7.

For the Decoding process we calculate the syndrome corresponding to the error at different bit positions and rectify the errors. The following values of syndrome are calculated corresponding to different error positions:

Received message Syndrome	Error	Syndrome	Received message	Error
10100100010101001100	no error	00000	101001010101010	pos8
00100100010101010011	pos1	01011	101001001101010	pos 9
11100100010101000110	pos2	11101	101001000001010	pos 10
10000100010101000111	pos3	11010	101001000111010	pos 11
10110100010101000101	pos4	10100	101001000100010	pos 12
10101100010101000001	pos5	01000	101001000101110	pos 13
10100000010101001001	pos6	11011	101001000101000	pos 14
10100110010101011001	pos7	10110	101001000101011	pos 15

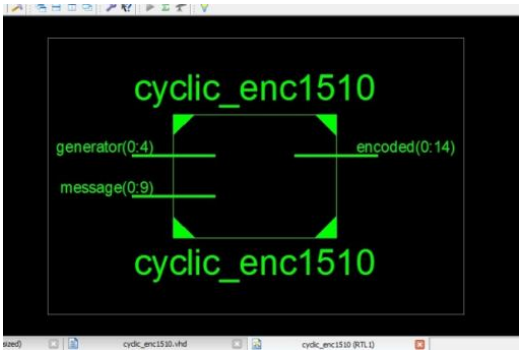

Fig. 3. Top Level Design of Encoder Circuit.

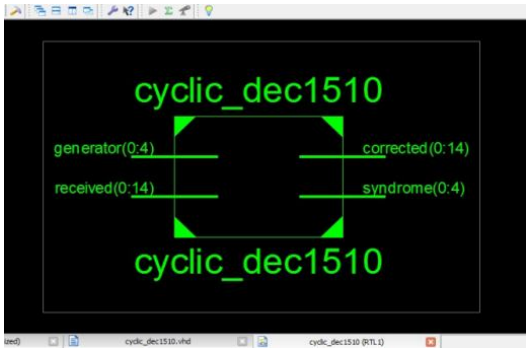

Fig. 4: Top Level Design of Decoder Circuit.

Fig. 3 and 4 shows the Top Level Design of the Encoder and Decoder Circuits Respectively.

Fig. 5 and 6 corresponds to the Gate Level Design of the Encoder and Decoder Circuits.

It can be seen that Gate Level Design of Decoder Circuit is more complex than that for Encoder Circuit.

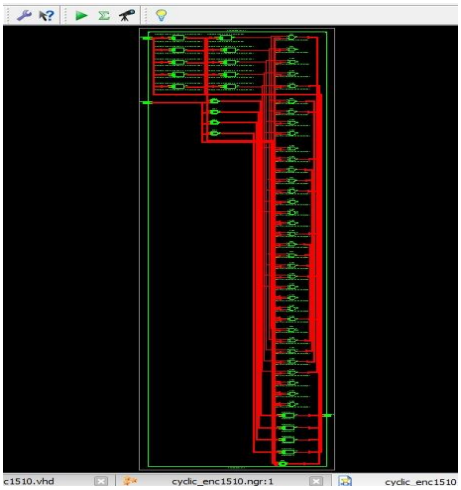

Fig. 5: Gate Level Design of Encoder Circuit.

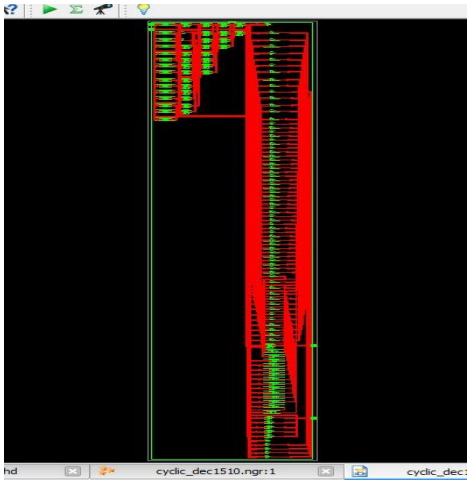

Fig. 6: Gate Level Design of Decoder Circuit.

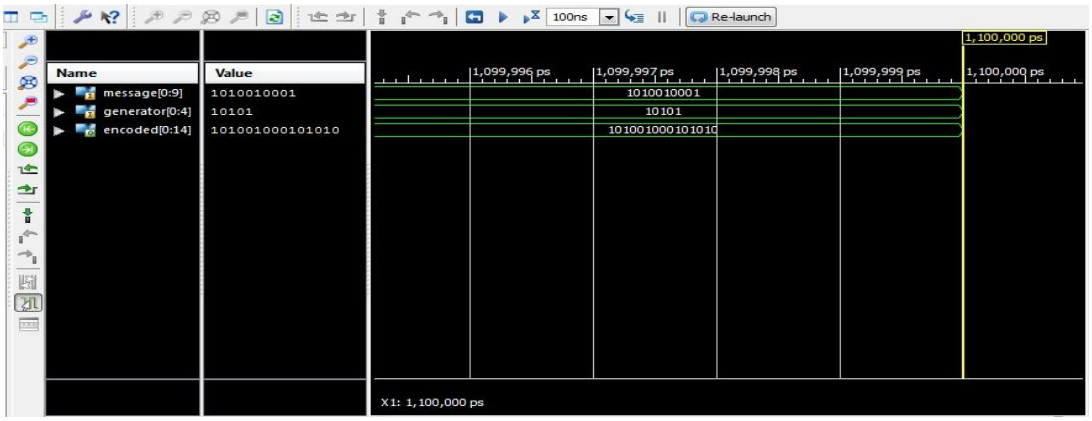

Fig. 7: Simulation of (15,10) Cyclic Code Encoder.

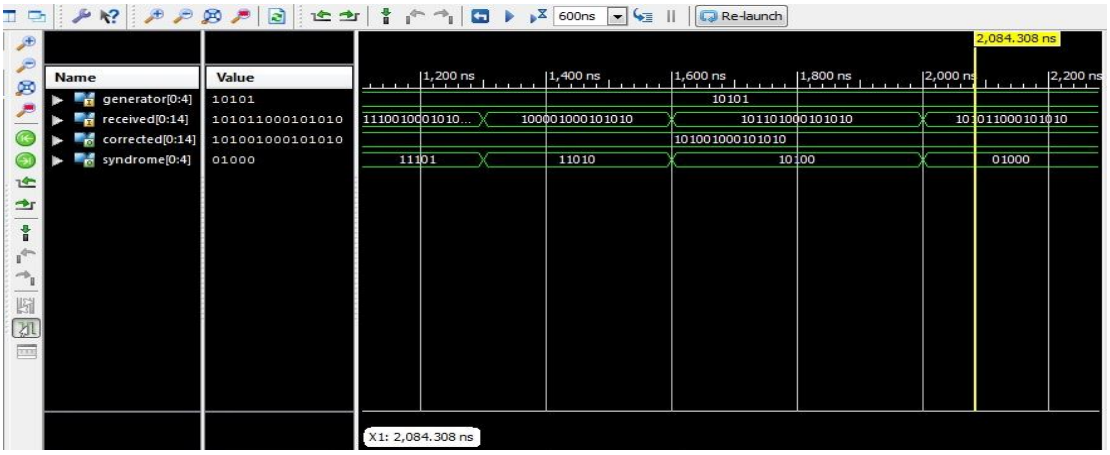

Fig. 8: Simulation of (15,10) Cyclic Code Decoder.

4. Hadware Implementation of Cyclic Code

To take into account the hardware constraints, we present here the hardware implementation of (7,4) Cyclic Code. Here we encode the input message 1101 with generator sequence 110. The encoded message comes out to be 1101001 with 001 as parity check bits. The message bits and the generator sequence are given as inputs to the FPGA kit using ON-OFF Switches as shown in figure. The encoded output is taken across the LEDs as shown.

Similarly the (7,4) decoder has been implemented and tested with different received inputs corresponding to noise at different bit positions. The corrected message appears across LEDs as shown.

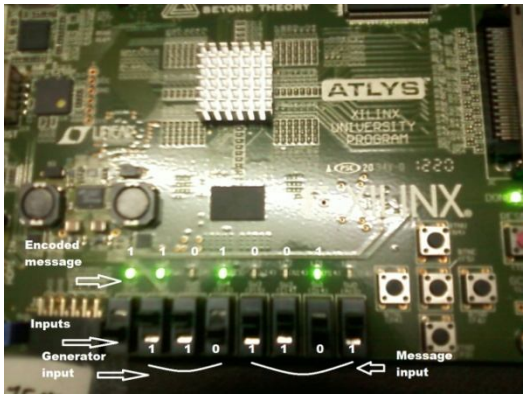

Fig. 9: FPGA implementation of (7,4) Encoder.

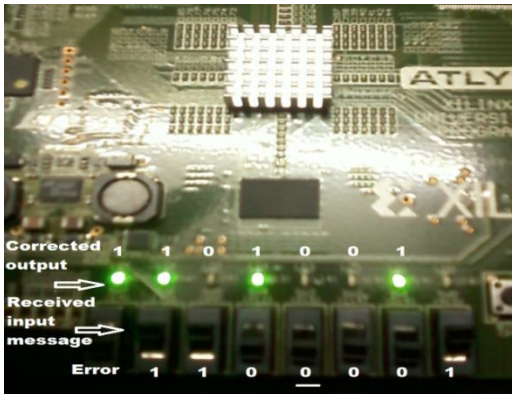

Fig. 10: FPGA implementation of (7,4) Decoder.

5. Conclusion

The VHDL design for Cyclic Code Encoder and Decoder has been simulated and implemented on hardware using Xilinx Spartan 6 LX45 FPGA mounted on Atlys board. The message bits were encoded and decoded both on software and hardware environments. The error bits were detected and successfully corrected both on hardware and software platforms. The VHDL design core both for Encoder and Decoder can be used to program the FPGA chips in accordance to the need in various applications.

References

[1] Wael M El-Medany, "FPGA implementation of CRC with Error Correction", ICWMC 2012: Eighth International Conference on Wireless and Mobile Communications, pp. 266-271

[2] Gaurav Bansal, Vishal Chaudhary.," VHDL implementation of (15,11) hamming encoder and decoder", National conference on advancements in the era of Multi-disciplinary systems , AEMDS –APRIL 2013, pp. 667-670.
[3] W.W Peterson, D.T Brown, IRE, 1961 "Cyclic Codes for Error Detection", pp.228-235
[4] Daniel F. Russell, " Cyclic Error Correcting Codes" December 14, 2001
[5] Shu Lin, Daniel J. Costello, JR., " Error Control Coding", Prentice Hall, New Jersey, 1983
[6] Digilent, " Atlys Board Reference Manual", October 28, 2010
[7] Jayaram Bhaskar " A VHDL Primer", AT & T Bell Laboratories, October 1991
[8] Kevin Skahill, "VHDL For Programmable Logic", Cypress Semiconductor, August 14, 1995
[9] Douglas Perry, " VHDL: Programming by Example", McGraw Hill Professional, 2002.