FPGA Implementation of CRC with Error Correction

Wael M El-Medany

Computer Engineering Department,
College of Information Technology,
University Of Bahrain, 32038 Bahrain,
Email: welmedany@uob.edu.bh

***Abstract* - This paper presents a Cyclic Redundancy Check (CRC) soft core design and its hardware implementation on Field Programmable Gate Array (FPGA). The core design includes both of the Encoder and Decoder systems to be used for the serial data transmission and reception of the Wireless Transceiver System. VHDL (VHSIC Hardware Description Language) has been used for describing the hardware of the Intellectual Property (IP) core chip. The core design has been simulated using and tested using ISim (VHDL/Verilog). Spartan 3A FPGA starter kit from Xilinx has been used for downloading the design into Xilinx Spartan 3A FPGA chip.**

Keywords-FPGA; CRC Code; IP Core; VLSI.

I. INTRODUCTION

In digital communication systems, the error detection is performed by computing checksum on the message that needs to be transmitted. The computed checksum is then concatenated to the end of the message to generate the codeword or the check sequence number to be transmitted. At the receiving end, the received word is compared with the transmitted codeword. If both are equal, then the message received is treated as error free, otherwise there is an error detected in the received word.

Cyclic Redundancy Check (CRC) Code has a wide range of applications in data communications and storage devices [1-6]. Cyclic Redundancy Check (CRC) is an error-checking block code that has been used for error detection only in which the received word has to be divided by a predetermined number called the generator number. If the reminder is zero, this means that there is no error detected, for nonzero reminder, this means that there is an error detected [7-10].

Cyclic Redundancy Check (CRCs) codes are so called because the check (data verification) code is a redundancy (it adds zero information) and the algorithm is based on cyclic codes [11]. CRC has applications also in Integrated Circuits Testing Design (ICTD), and Logical Fault Detections (LFD) [12]. In [3], Albertengo et al. derived a method for determining the logic equations for any generator polynomial. Their formalization is based on z-transform. To obtain logic equations, many polynomial divisions are needed. Thus, it is not possible to write a VHDL code that generates automatically the equations for CRC.

Normally, the design of the error control decoder is more complex than the encoder. CRC when first introduced was for error detection only, it can detect single bit error; burst error with length "w", where "w" equal to the number of bits for the Frame Check Sequence (FCS) number; and odd number of errors based on the value for the generator umber [14-15]. Further research has investigated theoretically a CRC with error correction capabilities. Shukla and Bergmann [1] failed to show their hardware implementation for CRC with one bit error correction, and the simulation for correcting the bit error.

The work presented in this paper describes the VHDL implementation of a CRC Decoder that has the advantages of correcting more than one bit error. Since we are introducing the hardware implementation for error CRC with error correction, our main concern is about the design of the CRC decoder with error correcting capabilities. The error correction in CRC decoder based on the error trapping technique, which is a cyclic linear block code [16-19]. Error trapping based on, cyclic shifting the received word on the division circuit, until the error can be trapped on the parity check bits. In that case the reminder will used as the error pattern, by which we can locate and correct the detected error.

The VHDL source code has been edited and synthesized using Xilinx ISE 13.1, and then simulated and tested using ISim (VHDL/Verilog). Spartan 3A FPGA starter kit from Xilinx has been used for downloading the design into Xilinx Spartan 3A FPGA chip. The design has been tested in a hardware environment for different data inputs.

The materials in this article are organized as follows: in Section II, a brief description of the State Machine (SM) chart of CRC encoding process; the SM chart for decoding algorithm is given in Section III; the modification in the decoding algorithm for error correction will be concluded in Section IV; in Section V, the circuit design for CRC decoder will be described, as well as the top-level design the decoder; the simulation results and discussion is given in Section VI; at the end, a conclusion will be given in Section VII.

II. SM CHART FOR CRC ENCODING PROCESS

The encoder generates an n-bit check sequence number from the given input k-bit information. The encoding process starts by calculation the Frame Check Sequence (FCS), by dividing the information bits by the predefined generator

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-203-5

number. The encoder then concatenates the FSC number to the k-bit information number to get the check sequence number with n-bits length. Figure 1 shows the SM chart for the encoding algorithm. Where m is k information bits, p is the generator number, and c is the check sequence number. In Figure 1, d and r are internal signals in VHDL architecture, where d has to be divided by r in order the get the reminder, which will be used as the FCS. The division process used in the encoding algorithm is the parallel division, which will be faster than the serial one. The serial division requires a number of clock cycles equal to the number of information bits in order to calculate the FCS; however the parallel one requires only one clock cycle.

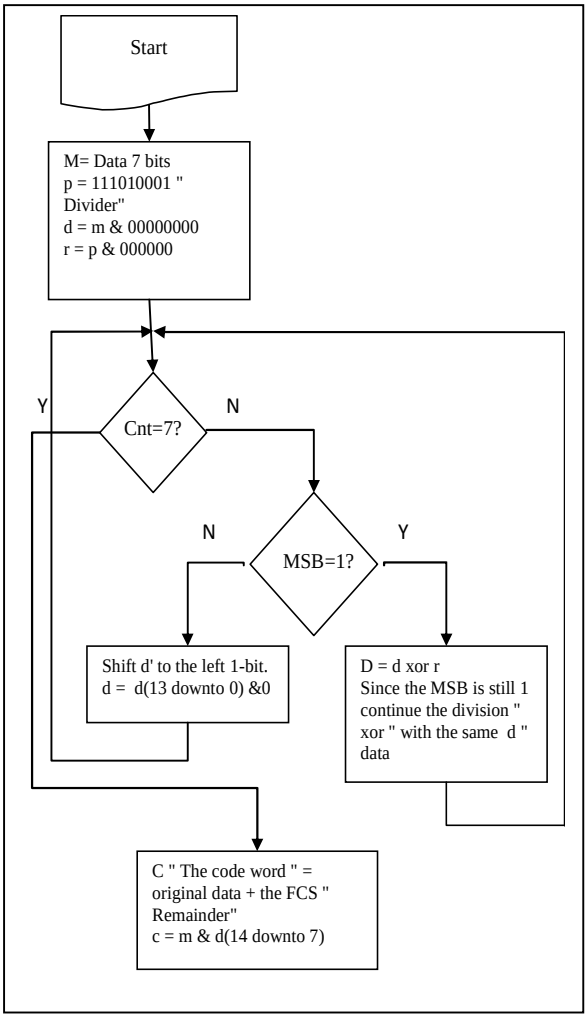

Figure 1. SM chart for CRC Encoding algorithm

III. SM CHART FOR CRC DECODING PROCESS

The CRC decoder is working similar to the encoder, where both of them based on using a division circuit. The decoding process starts by dividing the received check sequence number by the generator number. If the remainder is zero, assume there is no error detected. For nonzero remainder, it means that the received check sequence number got an error detected. The CRC decoder stop at the stage of detecting whether there is an error detected or not. But for our algorithm we are going to continue the division process until we get the error trapped, or get a decision that there is no error detected. As shown in Figure 2, the SM chart of the CRC decoding algorithm is similar to the one in Figure 1. However the decoder divide the received check sequence number 'c' by the generator number 'p', and check the reminder value whether it is zero or not, where zero reminders mean that there is no error detected, and none zero reminder means that there is an error detected.

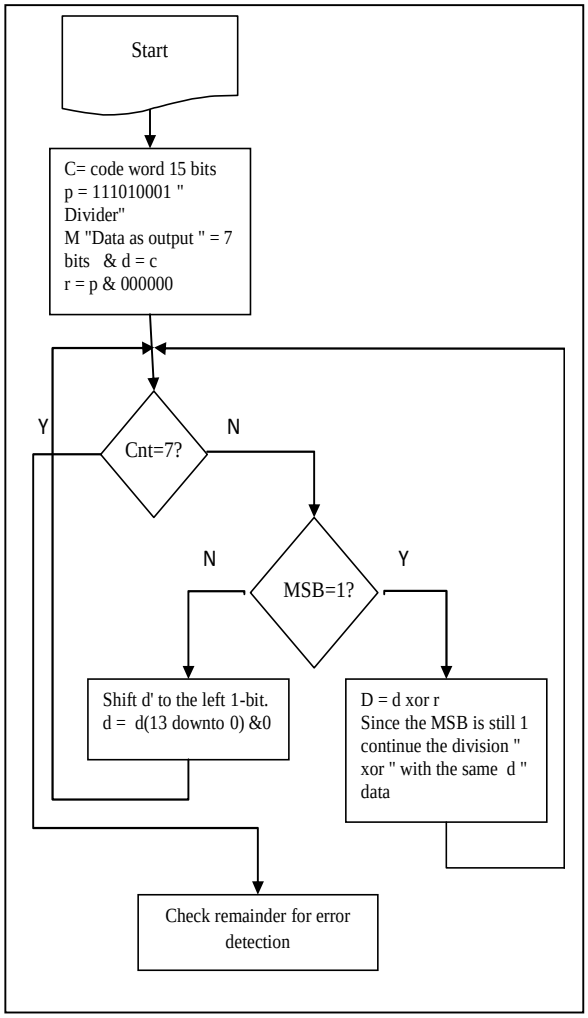

Figure 2. SM chart for CRC Decoding algorithm

IV. MODIFIED CRC DECODING ALGORITHM FOR ERROR CORRECTION

Cyclic Redundancy Check is a class of cyclic coding, which is one of the most powerful linear block codes. The modification for the CRC decoding algorithm based on the well known Error Trapping Technique (ETT), in which the

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-203-5

error has to be trapped in the parity bits. In CRC we will continue the division process until we get the error trapped in the FCS bits, or get a message that there is no error detected. This process can be done by cyclic shifting the generator number during the division process, and each time we do the division, we compare the reminder with number of errors that can be corrected using the linear block coding techniques, by calculating the Hamming distance for the code. Then use the following very famous equation:

$t = (d_{min} - 1)/2$

where t is the number of errors that can be corrected, and d_{min} is the minimum Hamming distance. The SM chart of the modified algorithm is shown in Figure 3, where the process of cyclic shifting has been added to the previous one shown in Figure 2, as well as checking the reminder after each division and compare it to the value of 't'.

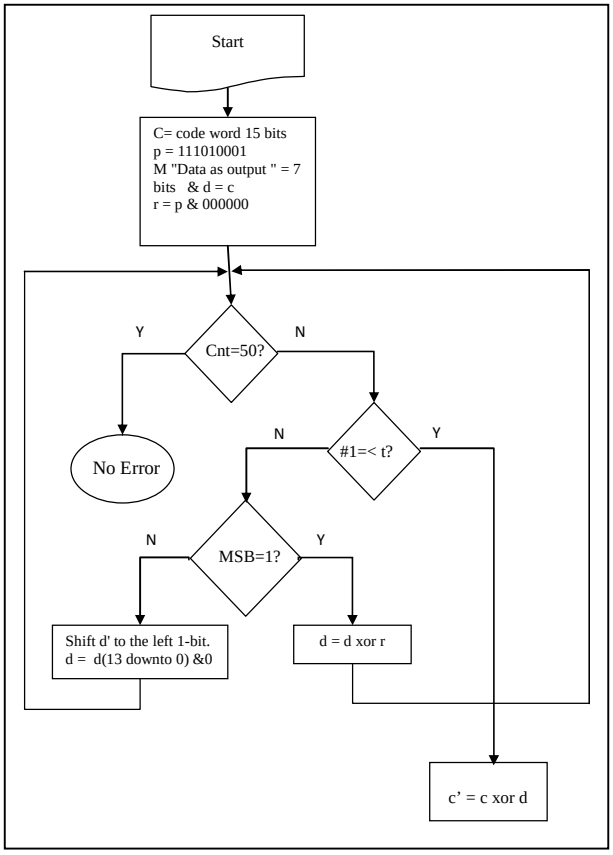

Figure 3. SM chart for a Modified CRC Decoding algorithm

If the number of non zero elements in the reminder is equal to or less than the value of 't', then the error has been trapped, and the reminder becomes the error pattern, which can be easily corrected. If this process would have been repeated a number of times, and the process has been entered in an infinite loop, then the process has to fished with the decision of "no error detected".

V. CRC Decoder Circuit Design

In this section, we are going to describe the hardware design for the CRC decoder circuit with error correction capability. The top-level design of the decoder circuit is given in Figure 4, which shows that the decoder has [Rx], in our example the received check sequence number [Rx] is 15-bit, which represent the code length, and the output of the decoder has 7-bits, which represent the information bits. In Figure 5, the second level of the top-down design is shown, the second level shows that there are three main units in the decoder circuit; the first unit is the Error Detection (ED) unit; the second unit is Locating Error (LE) unit; and the third one is the Error Correction (EC) unit. The top level of ED unit is given in Figure 6, which shows the inputs to this unit is [Rx], and the output is [sy] the syndrome that represents the remainder of the division process, and it is 8-bits, based on the values of the syndrome, whether it is zero or non zero, that will give indication whether there is an error detected or not.

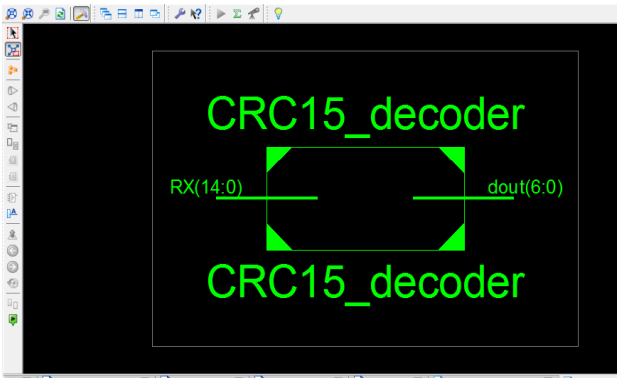

Figure 4. Top-level design of the decoder circuit

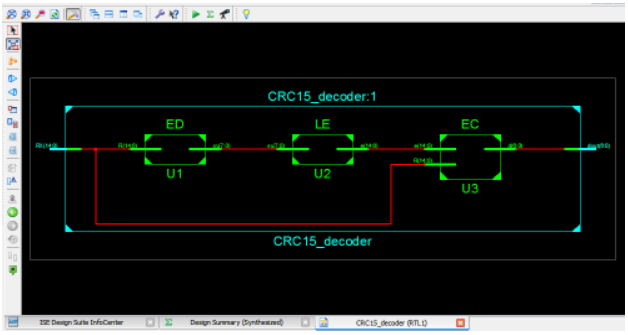

Figure 5. Second-level design of the decoder circuit

The register transfer logic for ED unit is given in Figure 7, in which there are 13 subunits, some of these subunits, we can see the gate level schematic, and some others are Xilinx FPGA building blocks. The top level of LE unit is given in Figure 8, which shows the inputs to this unit is [sy] coming from ED unit, and the output is [e] that represents the location of the bit in error. The register transfer logic for LE unit is given in Figure 9, which got only one building block

unit, called Mrom_e1, it is a ROM memory unit of the FPGA building blocks.

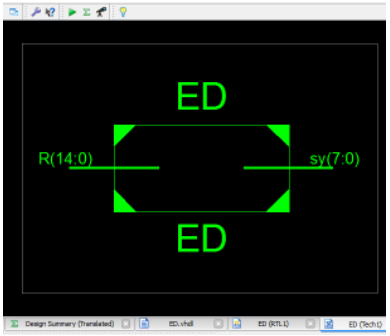

Figure 6. Top Level of the ED unit

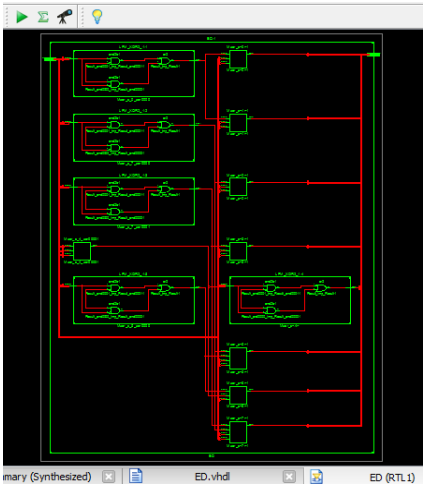

Figure 7. Register Transfer Level of the ED unit

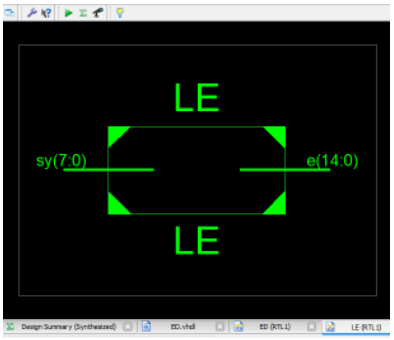

Figure 8. Top Level of the LE unit

The top level of EC unit is given in Figure 10, which shows the inputs to this unit is [R and e], and the output is [d] that represents the corrected data. The register transfer logic for EC unit is given in Figure 11, which got 7 building block units.

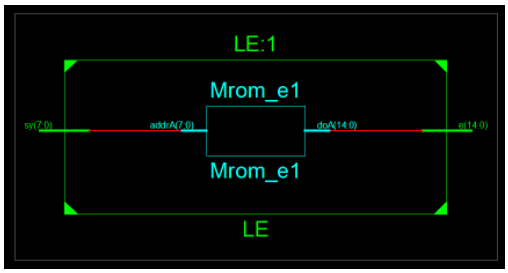

Figure 9. Register Transfer Level of LE unit

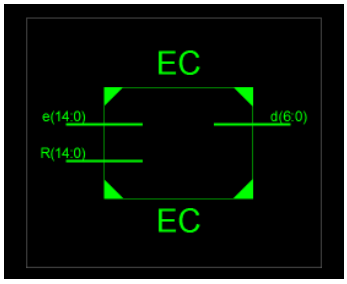

Figure 10. Top Level of the EC unit

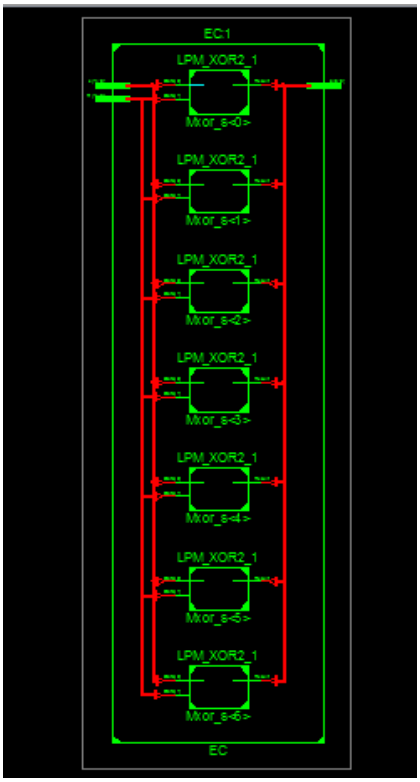

Figure 11. Register Transfer Level of the EC unit

VI. Simulation Results and Discussion

This section presents the results obtained through the simulation and design implementation summary are described. The presented CRC decoder can correct up to two errors. The gate level design of the error detection unit is given in Figure 12, and the gate level design of the error

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-203-5

correction unit is given in Figure 13. The test bench waveform for CRC decoder is given in Figure 14, with different inputs, and different errors. The simulation shown in Figure 14 for the modified CRC decoder and it can correct single bit error and double bit errors. For simplicity, we are given errors two codewords all one's codeword and all zero's codeword, "11111 11111 11111" and "00000 00000 00000". The inverted bits represent the introduced error, which is also represented by the signal [s2]. The signal [dout] is 7-bit corrected information, the simulation results for the presented CRC decoder proved the correctness of the decoder circuit either for error detection or error correction.

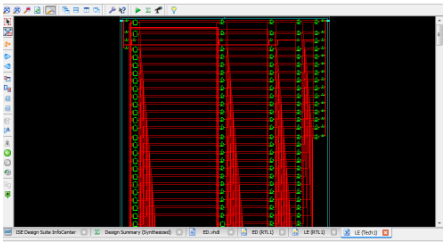

Figure 12. Gate Level Design of the ED unit

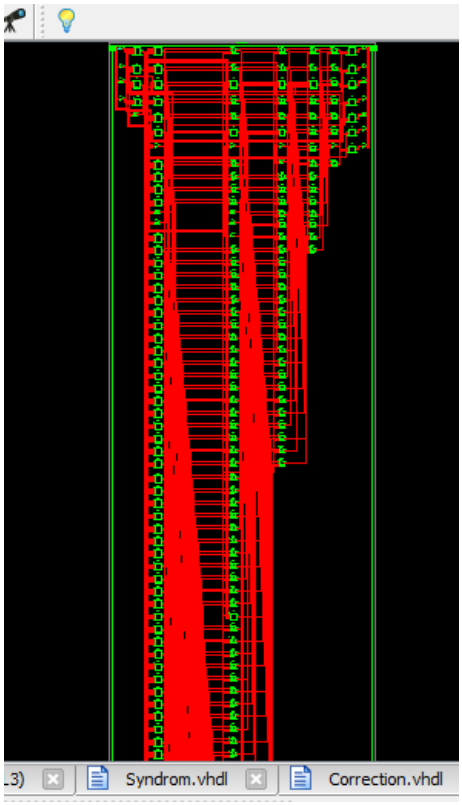

Figure 13. Gate Level Design of the EC unit

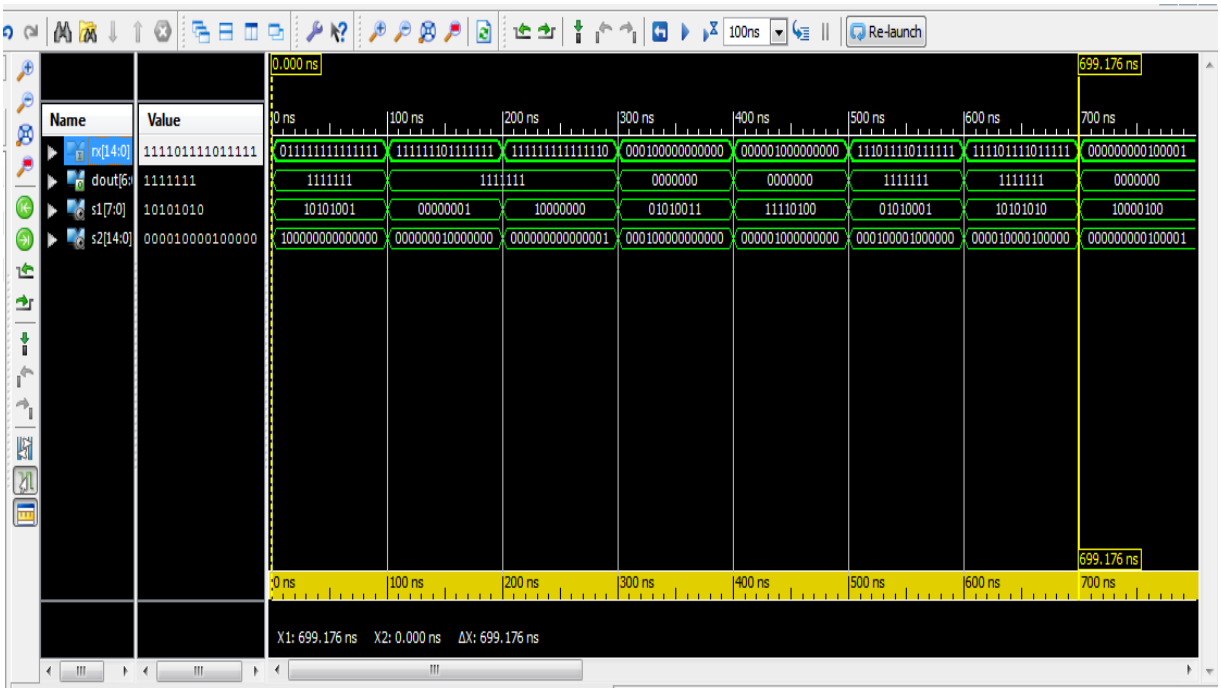

Figure 14. Test bench for CRC Decoder

VII. CONCLUSIONS

An FPGA design of a CRC decoder with error correction capabilities have been simulated and implemented. The system has been designed using VHDL, and implemented on hardware using Xilinx Spartan 3AN FPGA Starter kit. The CRC decoder for both error detection and error correction have been tested for different data inputs either for simulation purposes or in the hardware environment using the available FPGA kit. The VHDL source code has been edited and synthesized using Xilinx ISE 13.1, and then simulated and tested using ISim (VHDL/Verilog).

REFERENCES

[1] Shukla S. and Bergmann N. W., "Single bit error correction implementation in CRC-16 on FPGA", 1n: I EEE International Conference on Field-Programmable Technology. Brisbane, Australia, 2004. .

[2] W.W.Peterson and D.T.Brown, "Cyclic Codes for Error Detection", *Proc. IRE*, Jan. 1961.

[3] G. Albertengo and R. Sisto, "Parallel CRC Generation", *IEEE Micro*, Oct. 1990.

[4] R. Lee, "Cyclic Codes Redundancy," *Digital Design*, July 1977.

[5] A. Perez, "Byte-wise CRC Calculations", *IEEE Micro*, June 1983.

[6] A. K. Pandeya and T. J. Cassa, "Parallel CRC Lets Many Lines Use One Circuit", *Computer Design*, Sept. 1975.

[7] A. S. Tanenbaum, "*Computer Networks"*, Prentice Hall, 1981.

[8] T. V. Ramabadran and S. S. Gaitonde, "A tutorial on CRC computations", *IEEE Micro*, Aug. 1988.

[9] W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection", *Proc. IRE*, Jan. 1961.

[10] M. Sprachmann, "Automatic generation of parallel CRC circuits", IEEE Des. Test Comput., vol. 18, no. 3, pp. 108–114, May/Jun. 2001.

[11] N.R.Sexana and E.J.McCluskey, "Analysis of Checksums, Extended Precision Checksums and Cyclic Redundancy Checks", *IEEE Transactions on Computers*, July 1990.

[12] J.McCluskey, "High Speed Calculation of Cyclic Redundancy Codes," in *Proc. of the 1999 ACM/SIGDA seventh Int. Symp. on Field, 1999.*

[13] K. V. GANESH, D. SRI HARI, and M. HEMA, "Design and Synthesis of a Field Programmable CRC Circuit Architecture", International Journal of Engineering Research and Applications, ISSN 2248-9622**,** Volume 1, Issue 4, Nov-Dec 2011.

[14] William Stallins, "Data and Computer Communications", Eight edition, Prentice Hall, 2007.

[15] Behrouz A. Forouzan, "Data Communications and Networking", third edition, McGraw Hill, 2003.

[16] S. Lin and D. J. Cosetllo, "Error Control Coding: Fundamentals and Applications", Prentice Hall, NJ, 1983.

[17] R.E. Blahut, "Theory and Practice of Error Control Codes", Addison-Wesley, Menlo Park, California, 1983.

[18] G. Campobello, M. Russo, and G. Patanè, "Parallel CRC realization", IEEE Trans. Comput., vol. 52, no. 10, pp. 1312–1319, Oct. 2003.

[19] *Programmable Gate Arrays*, p. 250, ACM Press New York, NY, USA, 1999.

[20] G. Sharma‡, A. Dholakia□, and A. Hassan, "Simulation of Error Trapping Decoders on a Fading Channel", Proc. IEEE Vehicular Technology Conference, Atlanta, GA, 28 Apr.-1 May 1996, vol. 2, pp. 1361-1365

[21] M. J. S. Smith, "*Application-Specific Integrated Circuits"*, Addison-Wesley Longman, Jan. 1998.

[22] P.C. Hershey and C. B. Silio, "Finite State Machines for Information Collection and Assessment on High Speed Data Networks", Wireless and Optical Communications Proceeding, 2002.

[23] C. Borrelli, "IEEE 802.3 Cyclic Redundancy Check", application note: Virtex Series and Virtex-II Family, XAPP209 (v1.0), Xilinx, Inc, March 23, 2001.

[24] Efficient LDPC Decoder Implementation for DVB-S2 System, Apr 2010.

[25] D. Giot, P. Roche, G. Gasiot, and R. Harboe-Sorensen, "Multiple-Bit Upset Analysis in 90 nm SRAMs: Heavy Ions Testing and 3D Simulations", IEEE Trans. Nucl. Sci., Vol. 54, pp.904 – 911, Aug. 2007.

[26] Xilinx, "Spartan-3E Starter Kit Board User Guide", Xilinx, Tech. Rep. UG230, Mar 2006.

[27] Xilinx, "Virtex 5 Family Overview", Xilinx, Tech. Rep. DS100, Jun 2008.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-203-5