AN AUTOMATED CRC ENGINE

Mr. Hiren G. Patel, Dr. D.M.Patel, Mr. Milan A. Chaudhari, Mr. Mahavirsinh A. Zala

***Abstract*— The CRC or cyclic redundancy check is a widely used technique for error checking in many protocols used in data transmission. The main aim of this project is to design the CRC RTL generator or a tool that calculates the CRC equations for the given CRC polynomials and generates the Verilog RTL code .This block deals with the calculation of equations for standard polynomials like CRC-8, CRC-16, CRC-4, CRC-32 and CRC-64, CRC-32 and also user defined proprietary polynomial. Use PERL as the platform it also aims at having a simpler user interface and generate the RTLs for any data width and for any standard polynomial or user defined polynomial, and this design aims to be complete generic. RTLs generated by this tool are verified by System Verilog constrained random testing to make it more robust and reliable.**

***Index Terms*—** CRC-tools, HDL, PERL, RTL, VERILOG HDL.

I. INTRODUCTION

A CRC (Cyclic Redundancy Check) [1] is a popular error detecting code computed through binary polynomial division. To generate a CRC, the sender treats binary data as a binary polynomial and performs the modulo-2 division of the polynomial by a standard generator (e.g., CRC-32[2]). The remainder of this division becomes the CRC of the data, and it is attached to the original data and transmitted to the receiver. At receiver receiving the data and CRC, the receiver also performs the modulo- 2 division with the received data and the same generator polynomial. Errors are detected by comparing the computed CRC with the received one. The CRC algorithm adds a small number of bits (32 bits in the case of CRC-32) to the message regardless of the length of the original data, and shows good performance in detecting a single error as well as an error burst. Because the CRC algorithm is very good at detecting errors and is simple to implement in hardware, Today CRCs are widely used t for detecting corruption in digital data which may have occurred during transmission, Production or storage. And CRCs have recently found a new application in universal mobile telecommunications system standard for message length detection of variable-length message communications [3].

Traditionally, the LFSR (Linear Feedback Shift Register) circuit is implemented in VLSI (Very-Large-Scale Integration) to perform CRC calculation which can only process one bit per cycle [4]. In this project the method used for the generation of CRC polynomials is based on the LFSR CRC implementation, where the CRC is calculated by passing every and each data bit, feeding the most significant bit first and depending upon the data of the MSB register in the LFSR, shifting and XOR operations occur at every bit. This serial LFSR implementation is converted into a one shot or single cycle operation that is realized into a combinational circuit. On the Base of this method the CRC polynomials are generated.

II.CYCLIC REDUDANCY CHECK

A CRC is an error-detecting code. And its computation resembles a polynomial long division operation in which the quotient is discarded and the remainder becomes the result, with this important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field. The length of the remainder is less than the length of the divisor (called the generator polynomial), therefore determines how long the result can be. The definition of the particular CRC specifies the divisor to be used, among all other things.

The CRC is based on polynomial arithmetic system, in particular, on computing the remainder of dividing one polynomial in GF (2) (Galois field with two elements) by another polynomial. It is a little like treating the message as a very large binary number, and computing the remainder on dividing it by a fairly large prime like 2^32-5.Intuitively, one would expect this to give a reliable checksum. A polynomial in GF (2) is a polynomial in a single variable x whose coefficients are

Manuscript received June 17, 2014
Mr. Hiren G. Patel, Assistant Professor, EE, Bits Edu Campus, Vadodara, Gujarat, India
Prof. Dr. D.M.Patel, Professor and Head of Department, EE, Bits Edu Campus, Vadodara, Gujarat, India
Mr. Milan A. Chaudhari, Assistant Professor, EE, Bits Edu Campus, Vadodara, Gujarat, India
Mr. Mahavirsinh A. Zala, Assistant Professor, EE, Bits Edu Campus, Vadodara, Gujarat, India

0 or 1. After that Addition and subtraction are done modulo 2 – that is, they are same as the exclusive or (Ex-OR) operator. For an example, the sum of the polynomials:
$x^3 + x + 1$ and
$x^4 + x^3 + x^2 + x$
is $x^4 + x^2 + 1$, as is their difference. All these polynomials are not usually written with minus signs, but they could be, therefore a coefficient of –1 is equivalent to a coefficient of 1. Multiplication of those polynomials is straightforward. The main product of one coefficient by another coefficient is the same as their combination by the logical and operator, and the partial products are added using exclusive or (Ex-OR). Multiplication is not needed to compute the CRC checksum. Division of polynomials over GF (2) can be done in much the same way as long division of polynomials over the integers. Below is an example [5].

These might like to verify that the quotient of $x^4 + x^3 + 1$ multiplied by the divisor of $x^3 + x + 1$, plus the remainder of $x^2 + 1$, equals the dividend.

$$
\begin{array}{r}
x^4+x^3+1 \\
x^3+x+1\,\overline{)\,x^7+x^6+x^5+\qquad x^2+x} \\
\underline{x^7+\quad x^5+x^4} \\
\quad x^6+\quad x^4 \\
\quad \underline{x^6+\quad x^4+x^3} \\
\qquad\qquad x^3+x^2+x \\
\qquad\qquad \underline{x^3+\quad x+1} \\
\qquad\qquad\quad x^2+\quad 1
\end{array}
$$

The CRC method treats the message as a polynomial in GF (2). For example, the message 11001001, at where the order of transmission is from left to right (110…) is treated as a representation of the polynomial $x^7 + x^6 + x^3 + 1$.

Table 1: Generator polynomial of some CRC codes [5]

Common Name	r	Generator	
		Polynomial	Hex
CRC-12	12	$x^{12}+x^{11}+x^3+x^2+x+1$	80F
CRC-16	16	$x^{16}+x^{15}+x^2+1$	8005
CRC-CCITT	16	$x^{16}+x^{12}+x^5+1$	1021
CRC-32	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	04C11DB7

To develop the hardware circuit for computing the CRC checksum, we reduce the polynomial division process to its essentials.
The process employs a shift register, which is denoted by CRC. This is of length r (the degree of G) bits, not as you might expect. If the subtractions (exclusive or's) are done, than it is not necessary to represent the high-order bit, because the high-order bits of G. The division process might be described informally as follows [5]:

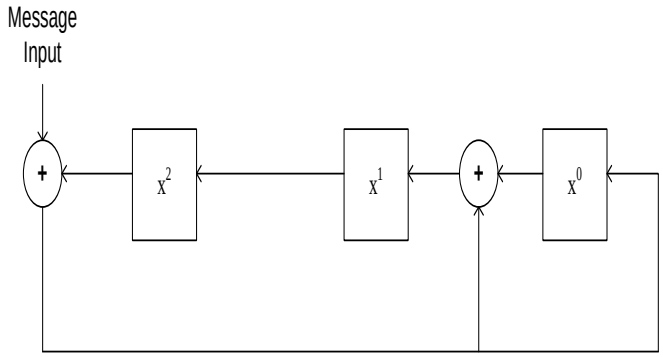

Figure 1: CRC circuit for $G=x^3 + x + 1$

III.IMPLEMENTATION OF CRC RTL GENERATOR

CRC Generator is a command line application that generates verilog code for CRC of any data width starts from 1 bit and no inherent upper limit and any standard polynomial or user defined polynomial. Code for this polynomial is written in Perl and is cross platform compatible for all. For verification, this tool provides CRC RTL generator which can be used at transmitter for CRC checksum generation and the receiver end.

The generated CRC module is very synthesizable verilog RTL tool. The method used for the generation of CRC polynomials is based on the LFSR CRC implementation, where the CRC is calculated by passing each and every data bit every cycle of data, feeding the most significant bit first.

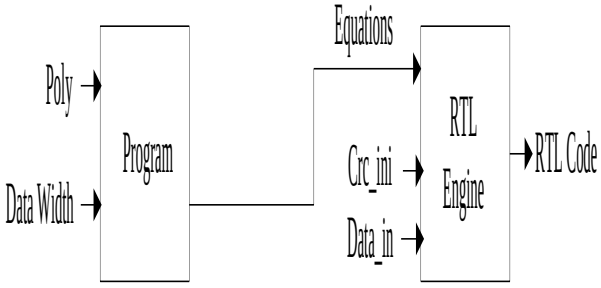

Figure 1: Block Diagram of CRC RTL Generator

Depending upon the data of the MSB (most significant bit) register in the LFSR, shifting and XOR operations take place. This serial LFSR implementation

is converted into a one shot or single cycle operation that is realized into a combinational circuit. The CRC polynomials are generated based on this method. Once all this RTL's of different polynomials are generated then the user can use these RTL's to calculate the CRC of entire packet.

IV.CRC PARAMETERS

Data width: Width of the data having ranges from 1 bit and no inherent upper limit.

Poly: Any user defined or Standard Polynomial.

Equations: The remainder of equations, these are the functions of data input and initial state remainders.

Data in: It is Input data to the Verilog code.

Crc_ini: It is Input initial remainder to the Verilog code.

RTL Engine: RTL Engine that generates the CRC bits.

Inputs-Polynomial: That is one among the above mentioned standard polynomials or a user defined proprietary polynomial.

Data Width:-Starting from 1 bit and no inherent upper limit.

Outputs - Verilog RTL code which in turn has its inputs as partial remainder, data (with the same width mentioned in the computational block) and output as final remainder.
RTL Engine - The RTL Engine takes Data stream, Initial Remainder as input and generates the RTL code as output using the equations from the Program block directly.

Program Block - The program block is the main block of the design part. It calculates the polynomial equations that help in building the XOR tree.

V.PLATFORM USED

PERL (Practical Extraction and Report Language): The major internal data structures in the Perl interpreter that represent Perl language elements. Our extractor interrogates the Perl internals just before the execution phase of the Perl script. At that moment the internal data structures are ready to be used for fact extraction. Perl is a compiling interpreter. Instead of interpreting line-by-line the script file, it reads the entire script file, converts it to an internal representation, and then executes the instructions [18].

PERL (Practical Extraction And Report Language) is used as platform for generating the RTL codes because of the constructs available .Perl had useful data structures like Hashes, Arrays, Hash of Arrays, Array of Hashes, which are very much useful in generating the polynomial equations.

VI.SIMULATION AND RESULTS

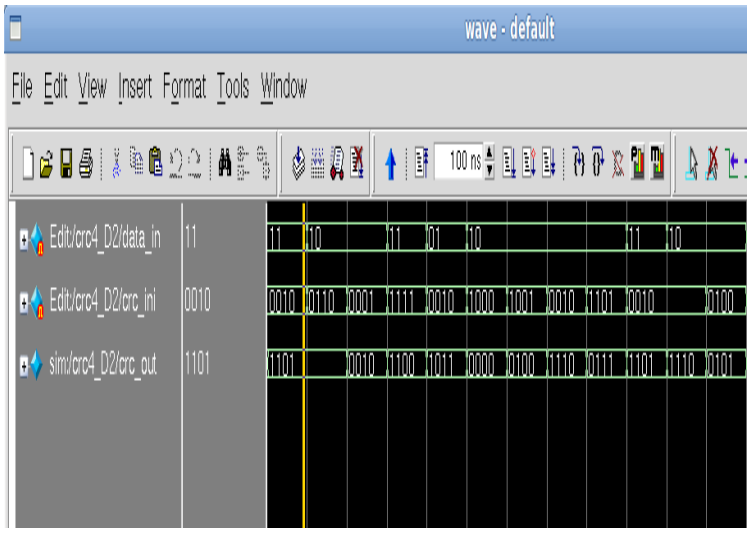

Fig. 3: Simulation results for CRC4 of data width is 2

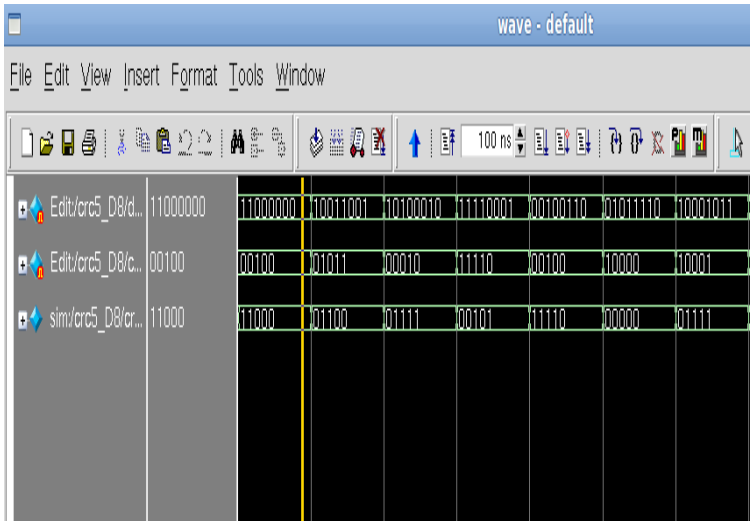

Fig. I: Simulation results for CRC5 of data width is 8

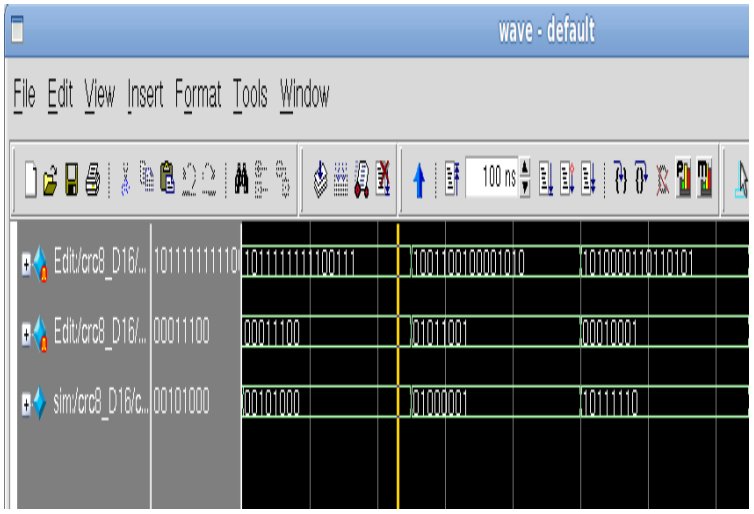

Fig. 5: Simulation results for CRC8 of data width is 16

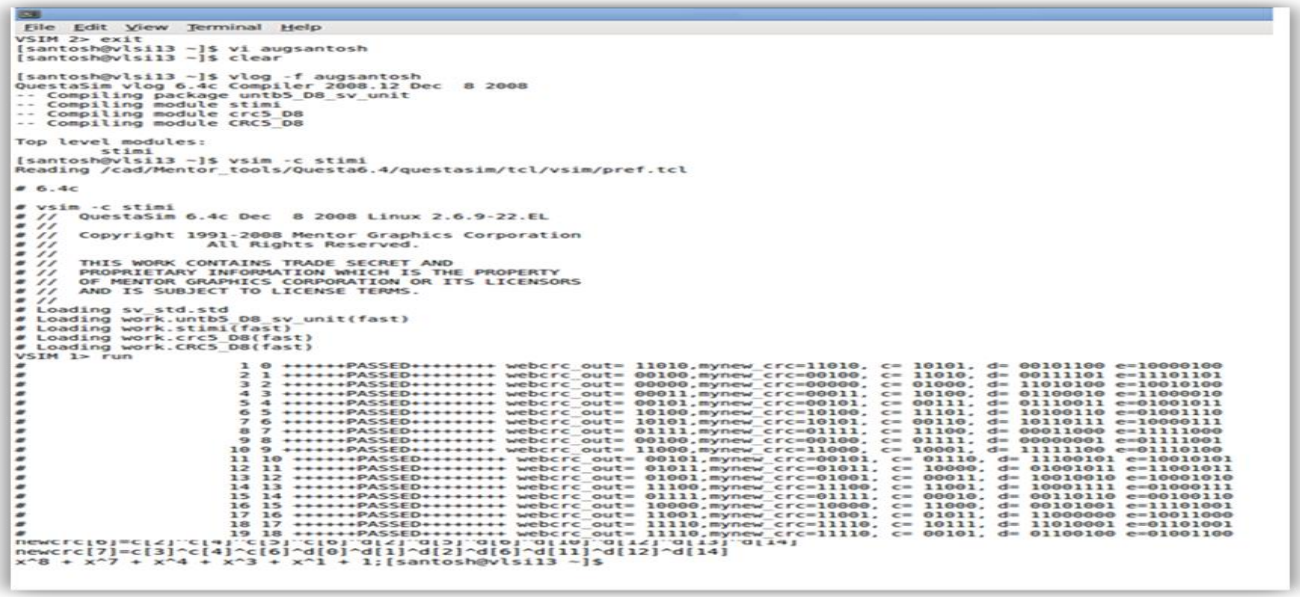

Fig. 6 Verification results of an augmented CRC RTL generator

VI.CONCLUSION

In this topic the necessary data for evaluation of the error control performance of CRC codes up to many bit redundancy is calculated. A very fast and easy to implement procedure for choosing the best CRC code is proposed. Codes of lengths greater than the order of the generator polynomial are considered and formula for the determination of the number of the code words of weight two is derived. We believe that the results obtained in this work will help designers of communication systems in selecting the most effective polynomial of degree as user definer for any particular application.The mainobjectives that is to design a tool that generates a Verilog RTL, that calculated the checksum for the given data polynomial and CRC polynomial on Perl and generating RTL for any data width and any polynomial.

To calculate the CRC equations for the given CRC polynomials designed a tool thatgenerates the Verilog code for any standard polynomials like CRC32, CRC24, CRC16, CRC8 and also any user defined polynomial and data width. The RTLs generated by this tool are verified by system Verilog constrained random verification to make it more robust and reliable. Hence the CRC applications are successfully Designed and verified

REFERENCES

[1] YAN SUN AND MIN SIK KIM," A TABLE-BASED ALGORITHM FOR PIPELINED CRC CALCULATION," IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC).PP 1-5, PUBLICATION YEAR 2010.

[2] K. BRAYER AND J. J. L. HAMMOND, "EVALUATION OF ERROR DETECTION POLYNOMIAL PERFORMANCE ON THE AUTOVON CHANNEL," IN CONFERENCE RECORD OF NATIONAL TELECOMMUNICATIONS CONFERENCE, VOL. 1, PP. 8–21 TO 8–25. 1975.

[3] S. L. SHIEH, P. N. CHEN, AND Y. S. HAN, "FLIP CRC MODIFICATION FOR MESSAGE LENGTH DETECTION," IEEE TRANSACTIONS ON COMMUNICATIONS,VOL. 55, NO. 9, PP. 1747–1756, PUBLICATION YEAR 2007.

[4] G. CAMPOBELLO, M. RUSSO, AND G. PATANÈ, "PARALLEL CRC REALIZATION", IEEE TRANS. COMPUT., VOL. 52, NO.10, PP. 1312–1319, OCT. 2003.

[5] HTTP://WWW.HACKERSDELIGHT.ORG/CRC.PDF - 2009-07-28 ACCESSED ON AUGUST/2012.

[6] QIAOYAN YU AND PAUL AMPADU," ADAPTIVE ERROR CONTROL FOR NOC SWITCH-TO-SWITCH LINKS IN A VARIABLE NOISE ENVIRONMENT," SIEEE INTERNATIONAL SYMPOSIUM ON DEFECTS AND FAULT TOLERENCEOF VLSI SYSTEMS, PP. 352 – 360, . PUBLICATION YEAR 2008.

[7] SHU LIN, DANIEL J. COSTELLO, JR. ERROR CONTROL CODING: FUNDAMENTALS AND APPLICATIONS. PRENTICE HALL. ISBN 0-13-283796-X.1983.

[8] HTTP://WWW.INTERLAKENALLIANCE.COM/INTERLAKEN_PROTOCOL_DEFINITION_V1.2.PDF ACCESSED ON SEPTEMBER/2012.

[9] HTTP://EN.WIKIPEDIA.ORG/WIKI/ERROR_DETECTION_AND_CORRECTIN.

[10] HEIDI JOKI, JARKKO PAAVOLA AND VALERY IPATOV "ANALYSIS OF REED-SOLOMON CODING COMBINED WITH CYCLIC REDUNDANCY CHECK IN DVB-H LINK LAYER,"2ND INTERNATIONAL SYMOPSIUM ON WIRELESS COMMUNICATION SYSTEMS , PAGE(S) 313 - 317 , PUBLICATION YEAR: 2005.

[11] T.V.; GAITONDE, S.S.; MICRO —"A TUTORIAL ON CRC COMPUTATION BY RAMABADRAN," MICRO IEEE ,VOL.: 8, ISSUE: 4: PAGE(S): 62 - 75. 1988.

[12] HTTP://EN.WIKIPEDIA.ORG/WIKI/CYCLIC_REDUNDANCY_CHECK ACCESSED ON JULY/2012.

[13] ROSS N. WILLIAMS, —A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS‖ VERSION: 3, DATE: 19 AUGUST 1993.

[14] /LINK/F CRC HTTP://WWW.REPAIRFAQ.ORG/FILIPGV32.HTML ACCESSED ON AUGUST/2012.

[15] HTTP://EN.WIKIPEDIA.ORG/WIKI/MATHEMATICS_OF_CRC#BITFILTERS. ACCESSED ON AUGUST2012.
[16] DANIEL L.MOISE KENNYWONG, "EXTRACTING FACTS FROM PERL CODE",REVERSE ENGINEERING WCRE'06, 13TH IEEE CONFERENCE, PAGES 1-10, PUBLICATION YEAR 2006.
[17] WWW.TESTBENCH.IN/CR 01 CONSTRAINED RANDOM VERIFICATION.HTML. ACCESSED ON APRIL 2011.
[18] HTTP://EN.WIKIPEDIA.ORG/WIKI/ERROR_DETECTION_AND_CORRECTION#CRYPTOGRAPHIC HASH FUNCTIONS. ACCESSED ON AUGUST 2012.
[19] TSONKA S. BAICHEVA —DETERMINATION OF THE BEST CRC CODES WITH UP TO 10-BIT REDUNDANCY.
[20] BEHROUZZOLFAGHARI, HAMEDSHEIDAEIAN, SAADAT POUR MOZAFFARI, "SYSTEMATIC SELECTION OF CRC GENERATOR POLYNOMIALS TO DETECT DOUBLE BIT ERRORS IN ETHERNET NETWORKS", 3RD INTERNATIONAL CONFERENCE ON COMPUTER NETWORKS & COMMUNICATIONS, ANKARA, TURKEY, 2011.
[21] SPRACHMANN, M.; , "AUTOMATIC GENERATION OF PARALLEL CRC CIRCUITS," DESIGN & TEST OF COMPUTERS, IEEE , VOL.18, NO.3, PP.108-114, MAY 2001.
[22] CAMPOBELLO, G.; PATANE, G.; RUSSO, M.; "PARALLEL CRC REALIZATION," COMPUTERS, IEEE TRANSACTIONS ON , VOL.52, NO.10, PP. 1312- 1319, OCT.2003.
[23] X. DENG, N. RONG, T. LIU, Y. YUAN AND D. YU, "SEGMENTED CYCLIC REDUNDANCY CHECK: A DATA PROTECTION SCHEME FOR FAST READING RFID TAG'S MEMORY," PROC. IEEE WCNC 2008, PP. 1576-1581, 2008.
[24] ZHANQI XU, AIJUN WEN, AND ZENGJI LIU, "SOME TRANSFORMS IN CYCLIC REDUNDANCY CHECK (CRC) COMPUTATION" , STATE KEY LAB ON INTEGRATED SERVICE NETWORK XIDIAN UNIVERSITY XI'AN, P. R. CHINA, 2011 IEEE.
[25] GLAISE R J. "A TWO-STEP COMPUTATION OF CYCLIC REDUNDANCY CODE CRC- 32 FOR ATM NETWORKS[J]". IBM JOURNAL OF RESERARCH AND DEVELOPMENT, 41(6): 700-710, 1997.
[26] G1AISE R J, JACQUART X. "FAST CRC CALCULATION". IN:PROCEEDINGS OF THE 1993 IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS & PROCESSORS. CAMBRIDGE, MA, USA, P-P 602-605, 1993.
[27] REHOBSON, K.L.CHEUNG, "A HIGH PERFORMANCE CMOS 32-BIT PARALLEL CRC ENGINE", IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL 40, NO. 2, FEB 1999.
[28] C. TOAL, K. MCLAUGHLIN, S. SEZER, AND XIN YANG. "DESIGN AND IMPLEMENTATION OF A FIELD PROGRAMMABLE CRC CIRCUIT ARCHITECTURE". VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE TRANSACTIONS ON, 17(8):1142 – 1147, AUG. 2009.
[29] A. AKAGIC AND H. AMANO,"AN FPGA IMPLEMENTATION OF CRC SLICINGBY- N ALGORITHMS". IEICE TECH. REP., VOL. 110, NO. 319, RECONF2010-42, PP. 19-24, NOV. 2010.
[30] C. CHENG AND K. K. PARHI, "HIGH-SPEED PARALLEL CRC IMPLEMENTATION BASED ON UNFOLDING, PIPELINING, AND RETIMING," IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS, VOL. 53, NO. 10, PP. 1017–1021, 2006
[31] A. AKAGIC AND H. AMANO,"HIGH SPEED CRC WITH 64-BIT GENERATOR POLYNOMIAL ON AN FPGA" THE INTERNATIONAL WORKSHOP ON HIGHLY EFFICIENT ACCELERATORS AND RECONFIGURABLE TECHNOLOGIES (HEART), 2-3 JUNE 2011, IMPERIAL COLLEGE, LONDON, UK.
[32] F. MONTEIRO, A. DANDACHE, A. M'SIR AND B. LEPLEY. "A FAST CRC IMPLEMENTATION ON FPGA USING A PIPELINED ARCHITECTURE FOR THE POLYNOMIAL DIVISION" IN PROC. OF THE 8TH IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, 2001.ICECS 2001, 1231 -1234 VOL.3.
[33] K. BRAYER AND J. J. L. HAMMOND, "EVALUATION OF ERROR DETECTION POLYNOMIAL PERFORMANCE ON THE AUTOVON CHANNEL," IN CONFERENCERECORD OF NATIONAL TELECOMMUNICATIONS CONFERENCE, VOL. 1, PP. 8–21 TO 8–25. 1975.
[34] QIAOYAN YU AND PAUL AMPADU, "ADAPTIVE ERROR CONTROL FOR NOC SWITCH-TO-SWITCH LINKS IN A VARIABLE NOISE ENVIRONMENT," IEEE INTERNATIONAL SYMPOSIUM ON DEFECTS AND FAULT TOLERENCEOFVLSI SYSTEMS, PP. 352 – 360,PUBLICATION YEAR 2008.
[35] SHU LIN, DANIEL J. COSTELLO, JR. "ERROR CONTROL CODING: FUNDAMENTALS AND APPLICATIONS".PRENTICE HALL. ISBN 0-13-283796-X, 1983.
[36] SAMIR PALNTIKAR. "VERILOG HDL-A GUIDE TO DIGITAL DESIGN AND SYNTHESIS". SUNSOFE PRESS, 1996, PAGE:6.
[37] W. W. PETERSON AND D. T. BROWN, "CYCLIC CODES FOR ERROR DETECTION," PROCEEDINGS OF THE IRE, VOL. 49, NO. 1, PP. 228–235, JAN. 1990.
[38] HEIDI JOKI, JARKKOPAAVOLA AND VALERY IPATOV"ANALYSIS OF REED-SOLOMON CODING COMBINED WITH CYCLIC REDUNDANCY CHECK IN DVB-H LINK LAYER," 2ND INTERNATIONAL SYMOPSIUM ON WIRELESS COMMUNICATION SYSTEMS, PAGE(S) 313 - 317, PUBLICATION YEAR: 2005.
[39] ROSS N. WILLIAMS, "A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS", VERSION: 3, DATE: 19 AUGUST 1993.
[40] DANIEL L.MOISEKENNYWONG, "EXTRACTING FACTS FROM PERL CODE", REVERSE ENGINEERING WCRE'06, 13TH IEEE CONFERENCE, PAGES 1-10, PUBLICATION YEAR 2006.