IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 3, Ver. II (May - Jun. 2015), PP 01-05
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Implementation of Parallel CRC Generation for High Speed Application

Payal.S.Hajare[1], Kanchan Mankar[2]

[1] *Research Scholar, Department of Electronics and Telecommunication Engineering, G.H.Raisoni Institute of Engineering and Technology for Women, Nagpur*
[2] *Assistant Professor, Department of Electronics and Telecommunication Engineering, G.H.Raisoni Institute of Engineering and Technology for Women, Nagpur*
Email: payalhajare238@gmail.com[1], kanchan.mankar@raisoni.net[2]

Abstract: *CRC is playing a main role in the networking environment to detect the errors. With challenging the speed of transmitting data to synchronize with speed, it is essential to increase speed of CRC generation. Most electronics engineers are familiar with the cyclic redundancy check (CRC).In the era of high speed data transmission, there requires a lot of accuracy for the message to be sent correctly to the receiver. And to check whether the message is correctly sent or not, an error detection technique called "CRC generation and check" is used. This technique informs the sender if any error is occurred. We know that it is widely used in communication protocols to detect bit errors and that it is essentially a remainder of the modulo-2long division operation. As a vital method for dealing with data errors usually the hardware implementation of CRC computations is based on the linear feedback shift registers (LFSRs), which handle the data in a serial way. As the serial calculation of the CRC codes cannot achieve a high throughput. Whereas in parallel CRC calculations can significantly increases the throughput of CRC computations. Mutants of CRCs are used in applications like CRC-16BISYNC protocols, CRC32 in Ethernet for error finding, CRC8 in ATM, CRC-CCITT in X-25 protocols, disk storage, SDLC, and XMODEM. This paper represents 64-bits parallel CRC architecture. The overall design is functionally simulated using Xilinx ISE Simulator.For the detection of the error in the received message; the CRC-bits are appended to the received message and then the 'Exclusive-OR'ing is performed by the polynomial. This gives us the generated CRC number, which is later used in the 'CRC check method', is used to verify the accuracy of data transmission.***

Keywords: *Cyclic redundancy check, Parallel CRC calculation, Shift register, Error control coding.*

I. Introduction

Cyclic Redundancy Check is a method adopted in the field of communication to detect errors during transmission through the communication channel. The data dispatched can be of any size depending on the type of data being transmitted. This paper describes a implemented VHDL code which demonstrates how the CRC process works on a codeword whose length can be changed by the user based on his requirements and the necessary simulations can be carried out to verify the results. Cyclic Redundancy Check (CRC) is an error detecting code in which a transmitted message is appended with a few redundant bits from the transmitter and then the codeword is checked at the receiver using modulo-2 arithmetic for errors. The data is sent from the encoder and is received by the receiver where a CRC check is carried out. This process helps to refer any errors in transmission through the transmission channel. The entire process is described by using **V**ery high speed Integrated Circuit **H**ardware **D**escription **L**anguage (**VHDL**). VHDL is a hardware description language used in electronic design automation to implement designs in systems such as field-programmable gate arrays.

Hitesh H. Mathukiya, Naresh M. Patel[1], Proposed the parallel CRC generation deal with 64bit parallel processing based on built in F matrix with order of generator polynomial is 32.The no. of LUT gets increased, so area also gets increase. Yan Sun; Min Sik Kim[2],Proposed a fast cyclic redundancy check (CRC) algorithm that performs CRC computation for an arbitrary length of message. Campobello, G.; Patane, G.; Russo, M.; [3],Proposed a theoretical result in the context of realizing high speed hardware for parallel CRC checksums. The number of bits processed in parallel can be different from the degree of the polynomial generator. Presented Pre-calculated F-matrix based 32 bit parallel processing which is faster and more compact and is independent of the technology used in its realization. But it doesn't work if polynomial change. Weidong Lu and Stephan Wong[4],Proposed presented a novel method to update the CRC code when packets are passing through interconnecting devices. It calculates the intermediate results of the changed fields based on the parallel CRC calculation and performs a single step update afterwards. And the number of cycles is dramatically reduced.The fast CRC update only calculates the changed portion of a frame.

This paper start with the introduction of CRC in section[1] followed by the introduction of serial and parallel crc in section[2]. The process of crc implementation in section[3], 64-bit Parallel Architecture in section[4], finally simulation results are shown in section[5] and concluded in section[6].

II. Generation of CRC

CRC can generate in two ways:
1. Serial CRC generation
2. Parallel CRC generation

Usually, the hardware implementation of CRC computations is based on the linear feedback shift registers (LFSRs), which handle the data in a serial way. Stills, the serial calculation of the CRC codes cannot give a high throughput. Hence, to over that defect we are moving to Parallel CRC generation.

1. Serial CRC generation:

Here CRC checking is done serially. The data input will be single (binary) and every clock pulse the data input will be one. There will some delay present between the consecutive data inputs and the output will be zero if the data will be encoded with same CRC value otherwise it shows non-zero value. The data is serially processed; the polynomial is XORed with the data input, that will be given to the D flip-flop (output same as the input) final CRC is generated as serially.

2. Parallel CRC generation:

Every modern communication protocol uses one or more error-detection algorithms. CRC is by far the most popular. CRC properties are interpreted by the generator polynomial length and coefficients. The protocol specification usually explains CRC in hex or polynomial notation. Parallel CRC calculation can significantly increases the throughput of CRC calculation.

III. Processes of CRC implementation

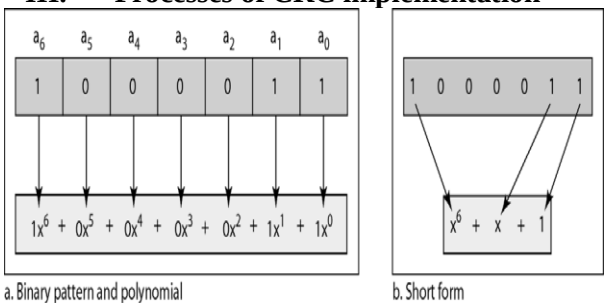

Fig 1: Method of polynomial detection

The technique of determining the polynomial is as follows:
Each value is considered as the coefficient of a particular term which is an exponent of x. The rightmost bit is contemplated as the 0^{th} and the next is the 1^{st} , then 2nd and continued.
For example, 1011 would mean a 4-bit polynomial.

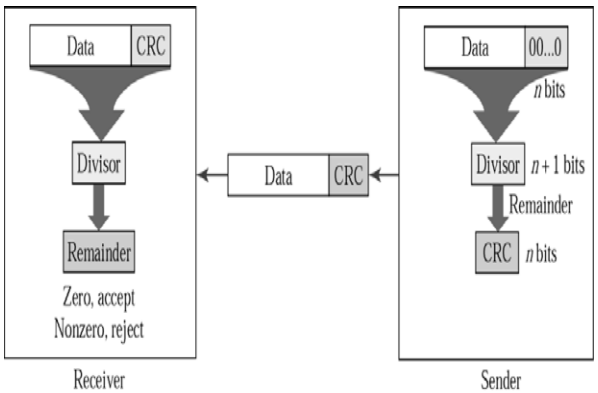

Fig 2: Block Diagram of Receiver and Sender

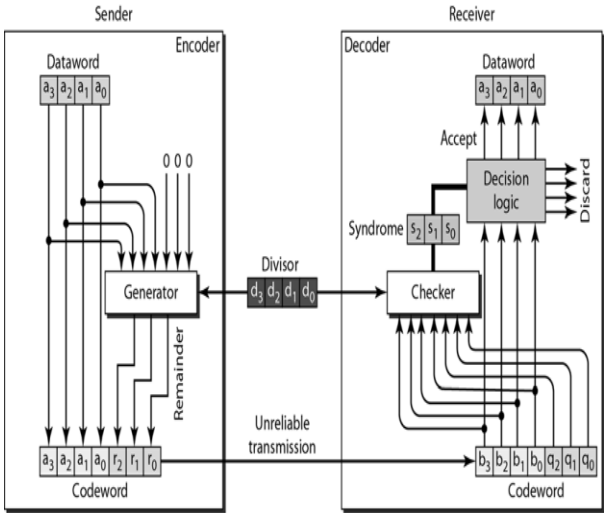

Fig 3: Bitwise Representation of the Encoder and Decoder

Considering a n-bit message is being transmitted and k is the number of data bits. Corresponding to the CRC process, a particular polynomial has to be selected and this polynomial is known as the divisor polynomial. The message is considered as the dividend and the divisor polynomial is used to divide the message polynomial to generate as a remainder. The method used for this purpose is known as modulo-2 division. In modulo – 2 division, carry bit in addition and borrow bit in subtraction generated from one particular bit is not carried forward to the next bit. In other way, for subtraction process simple XOR can perform the necessary operations. The message is inflated with (n-k) number of 0's. Then the modulo-2 division is processed and the remainder of (n-k) bits is generated. This remainder then replaces the (n-k) 0's at the end of the message sequence and it is then transmitted.

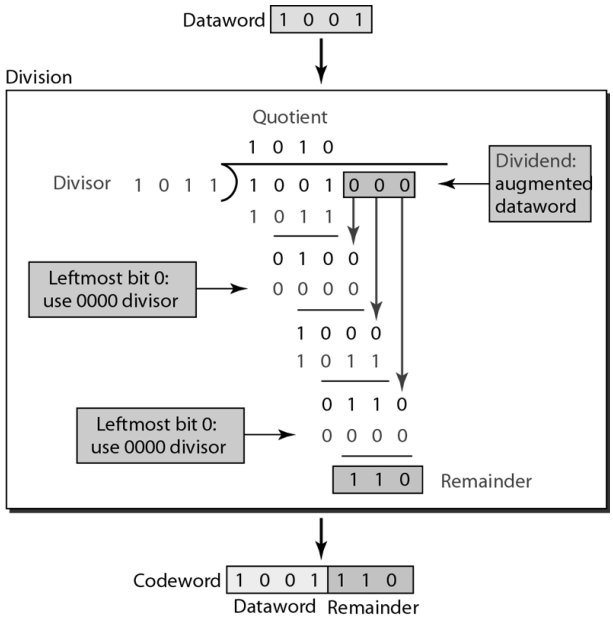

Fig 4: Division in CRC Generator

At the receiver, the data bits appended with the remainder is received as the data word. The data word is divided by the same generator polynomial to determine another remainder polynomial. If the polynomial obtained is 0, then it is known as error free. Otherwise the received message has errors. The entire process is seperated into two broad parts; ENCODER and DECODER. All the operations within the transmission of the dataword are carried out in the encoder while the checking operations are carried out in the decoder. For this reason, the encoder is also called as CRC Generator and the decoder is known as CRC checker.

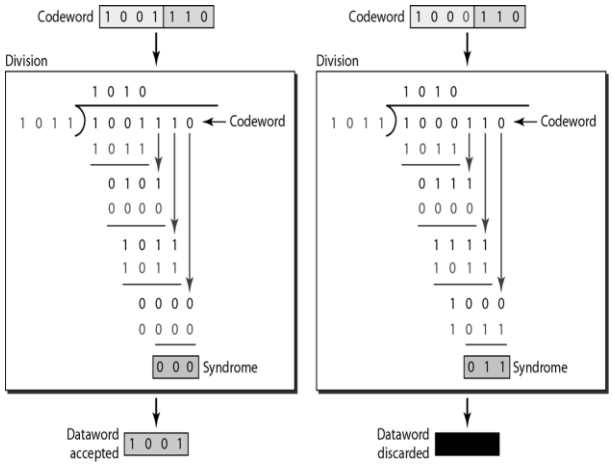

Fig 5: Division in CRC Checker with correct and erroneous codeword

IV. 64-bit Parallel Architecture

In proposed architecture CRC 64-bits are parallely processed and order of generator polynomial is 16. So, dramatically it can be concluded that if we increase the number of bits to be processed parallely, no of cycles required to calculate crc can be reduced. The methods applied to detect an error during transmission has been shown using simulation in VHDL. CRC is characterized by specification of a so called generator polynomial which is used as the devisor in a polynomial long division over a finite field taking the input data as dividend and where the remainder becomes the result. In 64-bit architecture, all 64-bits are dividend and for achieving error detection and correction is to add some redundancy (i.e. some extra data) to a message which receivers can use to check consistency of the delivered message. All 64-bits are divided by generator polynomial (16-bits) which is divisor and "Exclusive-ORing" is performed by the polynomial to get the final output of generated crc. For checking process, this generated crc bits are again attached with 64-bits the Ex-ORing operation takes place with same order of generator polynomial. If the output (remainder) is zero that means it is error free. If the output is not zero an error has occurred during the transmission process.

V. Simulation results

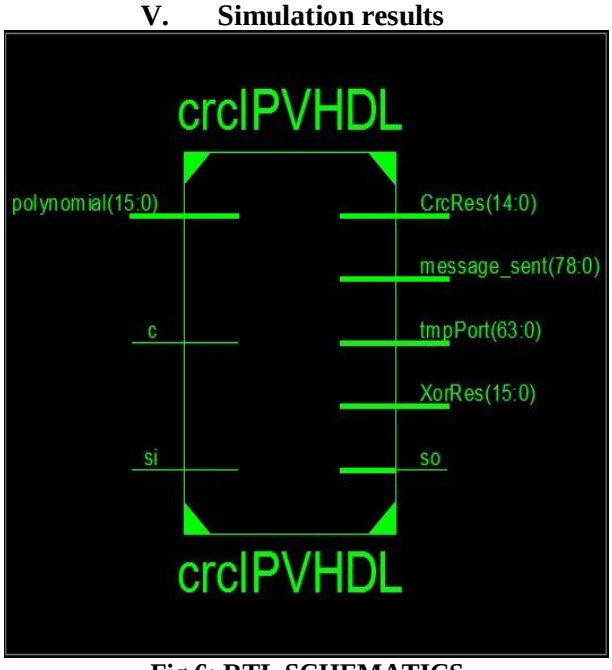

Fig 6: RTL SCHEMATICS

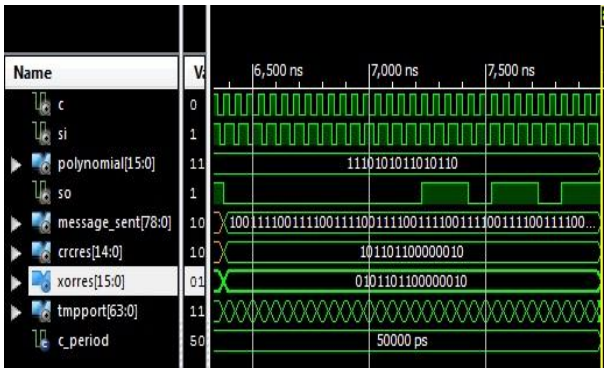

Fig 7: Generated Waveform for proposed 64-bit Architecture.

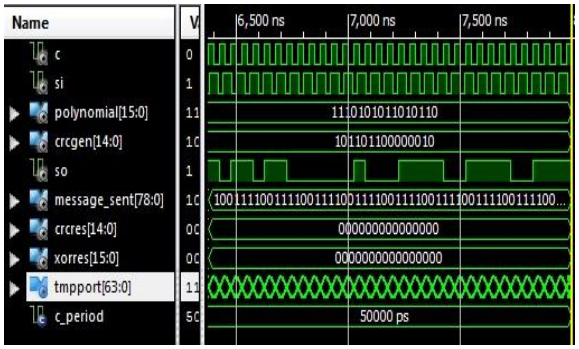

Fig 8: Checker Waveform for 64-bit Architecture

VI. Result

For the detection of the error in the receive message, the CRC bits are appended to the received message and then the 'Exclusive-ORing' is performed by the polynomial. From this we obtain the generated CRC number, which is used in the 'CRC check method', to examine the accuracy of data transmission. At the decoder, we see that the remainder is equal to a string of 0's, indicating that if the received codeword is same as that of the transmitted codeword in the encoder, that means it is error free. Thus, the received codeword has been intentionally made to be equal to the transmitted codeword to show a successful transmission.

VII. Conclusion

From the related survey it comes to know that when high speed data transmission is required serial implementation is not preferred because of slow throughput. So, parallel implementation is adopted which does not takes much time. CRC gives trade of between flexibility, performance and cost has been taken further than those enabled by tradition heterogeneous architectures based on microprocessor, DSP. CRC is very useful for error detection during data transmission. Having a look on the "CRC error detection method", we can prefer that the method identifies the error correctly. Thus the 'Parallel CRC generation and checking' is more efficient than the 'Serial CRC generation', it gives us the efficient result.

References

[1]. Hitesh H. Mathukiya, "A Novel Approach for parallel CRC generation for high speed application, "International conference on communication system and network technologies, no. 978-0-7695-4692-6/12-2012 IEEE.

[2]. Yan Sun; Min Sik Kim. "A Pipelined CRC Calculation Using Lookup Tables," Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE , vol., no., pp.1-2, 9-12 Jan.2010

[3]. G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," IEEE Transactions on Computers, Oct. 2003.

[4]. Weidong Lu and Stephan Wong, "A Fast CRC Update Implementation", IEEE Workshop on High Performance Switching and Routing, Oct. 2003.

[5]. Tom´a˘s Z´avodn´ık, Luk´a˘s Kekely, Viktor Pu˘s, "CRC Based Hashing in FPGA Using DSP Blocks" at 978-1-4799-4558-0/14/$31.00 ©2014 IEEE

[6]. A.K. Pandeya and T.J. Cassa, "Parallel CRC Lets Many Lines Use One Circuit," Computer Design, Vol. 14, No.9, Sept. 1975, pp. 87-91.

[7]. R. Lee, "Cyclic Code Redundancy," Digital Design, Vol.11, No. 7, July 1977, pp. 77-85.

[8]. G. Albertengo and R. Sisto, "On the Implementation of an Interface for a Multichannel Local Area Network," Proc. Melecon, IEEE, Apr. 1989, pp. 649-653.

[9]. Jing-Shiun Lin, Chung-Kung Lee, Ming-Der Shieh, and Jun-Hong Chen, "High-Speed CRC Design for 10 Gbps Applications" National Science Council. 0-7803-9390-2/06/$20.00 ©2006 IEEE

[10]. Christopher Kennedy and Arash Reyhani-Masoleh, " High speed Parallel CRC Circuit" Asilmar 978-1-4244-2941-7/08©2008 IEEE