A NOVEL DESIGN AND SIMULATION OF CYCLIC REDUNDANCY CHECK ENCODER AND DECODER

L. Prasad Naik[1], M. Suguneswari[2], P. Chandana[3]
[1]*Assistant Professor,* [2]*Assistant Professor,* [3]*Assistant Professor, ECE Department, Anantha Lakshmi Institute of Technology and Sciences, Ananthapuramu, Andhra Pradesh, India.*

In this Paper , A technique to extend the 3-bit BEC codes with the QAEC is presented. The proposed codes have the same redundancy as the previous 3-bit BEC codes . To accelerate the searching process of target matrices, a new algorithm with column weight control and recording function is proposed. Based on the proposed algorithm, a searching tool is developed to execute the searching process automatically. To prove the validity of the proposed algorithm, it is applied to the previous 3-bit BEC codes and the codes are remarkably improved on the two optimization criteria The use of error-correction codes (ECCs) with advanced correction capability is a common system-level strategy to harden the memory against multiple bit upsets (MBUs). Therefore, the construction of ECCs with advanced error correction and low redundancy has become an important problem, especially for adjacent ECCs. Existing codes for mitigating MBUs mainly focus on the correction of up to 3-bit burst errors. As the technology scales and cell interval distance decrease, the number of affected bits can easily extend to more than 3 bit. The previous methods are therefore not enough to satisfy the reliability requirement of the applications in harsh environments. The encoding and decoding procedure of the proposed codes is illustrated with an example. Then, the encoders and decoders are implemented using a 65-nm library and the results show that our codes have moderate total area and delay overhead to achieve the correction ability extension.

I.INTRODUCTION

Most of communication system protocols use one or more method to detect errors, so to correct it, by retransmission data or using Forward Error Correction(FEC) methods. One of the most commonly method in digital communication system to detect error is Cyclic redundancy check (CRC), which used in digital network or at storage device to detect errors. CRC codes is based on cyclic error correction codes theorem which is used systematic codes to encode the original data (message) for error detection at the receiver side.

This method was Invented by W. Wesley Peterson, and published in 1961 ,which is a type of linear block codes deals with systematic error detecting code used a group of error control bits appended to the

end of the message block. The error control bits represent the remainder of a divisionbetween the original message and the generator polynomial. The importance of CRC method came from its burst-error detection capability and all single error with an arbitrary data (message) length. The received message pass through a sequence of operations to detect if it has an error or not. The receiver has the ability to send retransmission requests back to the data source through a feedback channel. The transmitter retransmit correct data again or other methods like hamming code is added to correct errors by the receiver directly.

Thus, as stated, it is obvious the importance of CRC in communication systems so, in this paper efficient CRC encoder and decoder circuits are designed and simulated using ISE (Integrated Software Environment) Xilinx Design Suite. The proposed circuits can be used to detect errors for any input data size. Also, hamming code error correction method can be added to the proposed design easily. Cyclic codes have wide use in communication systems since they exhibit good error detecting and correcting properties. Cyclic Redundancy Check and Bose Chaudhuri Hocquenghem codes are the least difficult and most regularly used cyclic codes. In serial CRC structure, the computational time is equal to k clock cycles, where k is the no of message bits. By adopting such a parallel architecture, CRC can be calculated in k/m clock cycles, where m is the no of inputs given into the architecture at a time. Such an architecture can increase the throughput and allows high speed communication. But, the increase in hardware cost and longer critical path leads to area complexity.

The method of determining the polynomial is as follows: Each value is considered as the coeffiecient of a particular term which is an exponent of x. The rightmost bit is considered as the 0th, the next is the 1st, then 2nd and so on. For example, 1011 would mean a polynomial of $[(1 * x^0) + (1 * x^1) + (0 * x^2) + (1 * x^3)] = x^3 + x + 1$ (starting from rightmost).

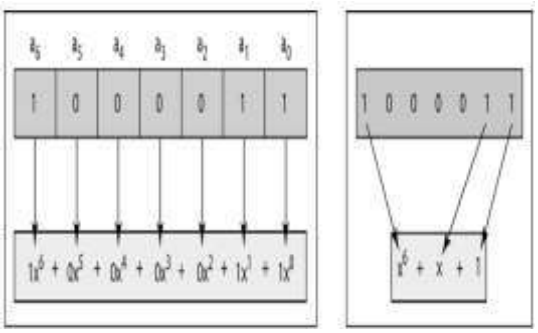

Fig1.1: Method of polynomial detection and its short form

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. If an error

occurred, the receiver sends a "negative acknowledgement" (NAK) back to the sender, requesting that the message be retransmitted. The technique is also sometimes applied to data storage devices, such as a disk drive. In this situation each block on the disk would have check bits, and the hardware might automatically initiate a reread of the block when an error is detected, or it might report the error to software. It consists of a single exclusive or gate together with some control circuitry. For bit parallel transmission, an exclusive or tree may be used, to compute the parity bit in software shown in fig 2.

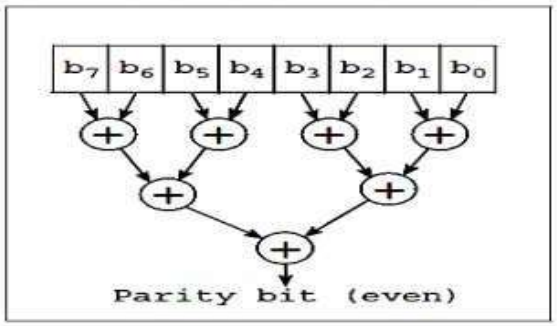

Fig1.2 :Exclusive or tree

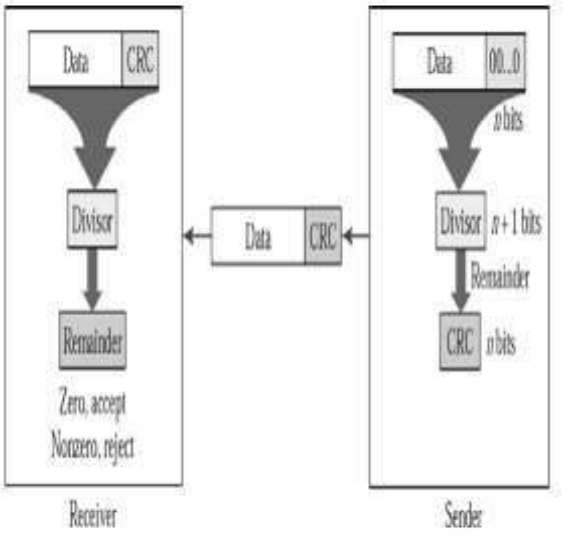

Fig 1.3:Block Diagram of Receiver and Sender in Cyclic redundancy Check

II. LITERATURE SURVEY

Error correction codes provide a means to detect and correct errors introduced by a transmission channel. Two main categories of code exist: block codes and convolution codes. They both introduce redundancy by adding parity symbols to the message data.

Cyclic redundancy check (CRC) codes are a subset of cyclic codes that are also a subset of linear block codes. The theory behind block coding and more specifically CRC coding is briefly discussed in this application report as well as most common CRC codes. CRC implementation can use either hardware or software methods. In the traditional hardware implementation, a simple shift register circuit performs the computations by handling the data one bit at a time. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted.

Dr. Hamming in 1947 first time introduces the idea of error-correcting codes and he constructed first error correcting code namely "Hamming code". Later he published a paper [14] on error detecting and error correcting codes in Bell System Technical Journal. C. E. Shannon in 1948 introduced the Mathematical Theory of Communication and published a paper [7] in Bell System Technical journal. M. J. E. Golay in 1949 published an article [15] on digital coding in the Proceedings of the IEEE and constructed a (23, 12) Golay code. Dr. Reed [16] and Dr. Muller [17] in 1954 described a new error correction code in their report on logic design.

Dr. Reed also gave the ring theory of algebraic [18], Galois field and polynomials [19] over fields.

III. EXISTING METHOD

CRC is more powerful than VRC and LRC in detecting errors. It is not based on binary addition like VRC and LRC. Rather it is based on binary division.At the sender side, the data unit to be transmitted IS divided by a predetermined divisor (binary number) in order to obtain the remainder. This remainder is called CRC.The CRC has one bit less than the divisor. It means that if CRC is of n bits, divisor is of n+ 1 bit.The sender appends this CRC to the end of data unit such that the resulting data unit becomes exactly divisible by predetermined divisor *i.e.* remainder becomes zero.At the destination, the incoming data unit *i.e.* data + CRC is divided by the same number (predetermined binary divisor). If the remainder after division is zero then there is no error in the data unit & receiver accepts it. If remainder after division is not zero, it indicates that the data unit has been damaged in transit and therefore it is rejected. This technique is more powerful than the parity check and checksum error detection.CRC is based on binary division. A sequence of redundant bits called CRC or CRC remainder is appended at the end of a data unit such as byte.

3.2 REQUIREMENTS OF CRC:

A CRC will be valid if and only if it satisfies the following requirements:

1. It should have exactly one less bit than divisor.
2. Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

3.3 THE VARIOUS STEPS FOLLOWED IN THE CRC METHOD ARE

1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.
2. The newly formed data unit *i.e.* original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder is called CRC.

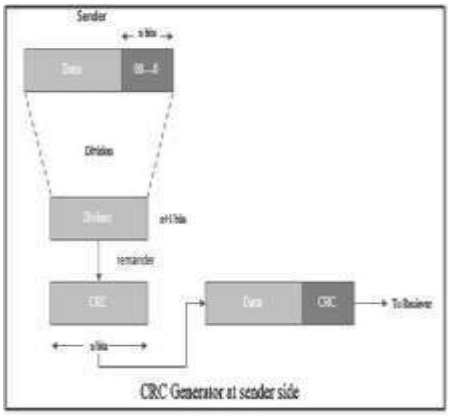

Fig: 3.1 CRC Generator at Transmitter side

Now, string of n Os appended to data unit is replaced by the CRC remainder (which is also of n bit).

3. The data unit + CRC is then transmitted to receiver.
4. The receiver on receiving it divides data unit + CRC by the same divisor & checks the remainder.

5. If the remainder of division is zero, receiver assumes that there is no error in data and it accepts it.
6. If remainder is non-zero then there is an error in data and receiver rejects it.

- For example, if data to be transmitted is 1001 and predetermined divisor is 1011. The procedure given below is used:

1. String of 3 zeroes is appended to 1011 as divisor is of 4 bits. Now newly formed data is 1011000.

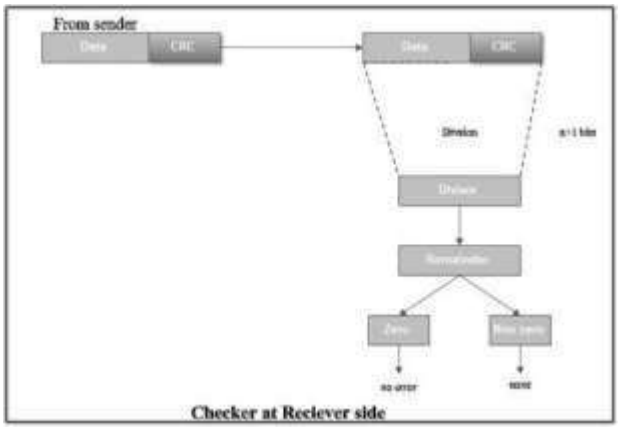

Fig: 3.2 CRC Generator at Receiver side

Data unit 1011000 is divided by 1011.

2. During this process of division, whenever the leftmost bit of dividend or remainder is 0, we use a string of Os of same length as divisor. Thus in this case divisor 1011 is replaced by 0000.
3. At the receiver side, data received is 1001110.
4. This data is again divided by a divisor 1011.
5. The remainder obtained is 000; it means there is no error.

CRC can detect all the burst errors that affect an odd number of bits.

- The probability of error detection and the types of detectable errors depends on the choice of divisor.
- Thus two major requirement of CRC are:

(a) CRC should have exactly one bit less than divisor.

(b) Appending the CRC to the end of the data unit should result in the bit sequence which is exactly divisible by the divisor.

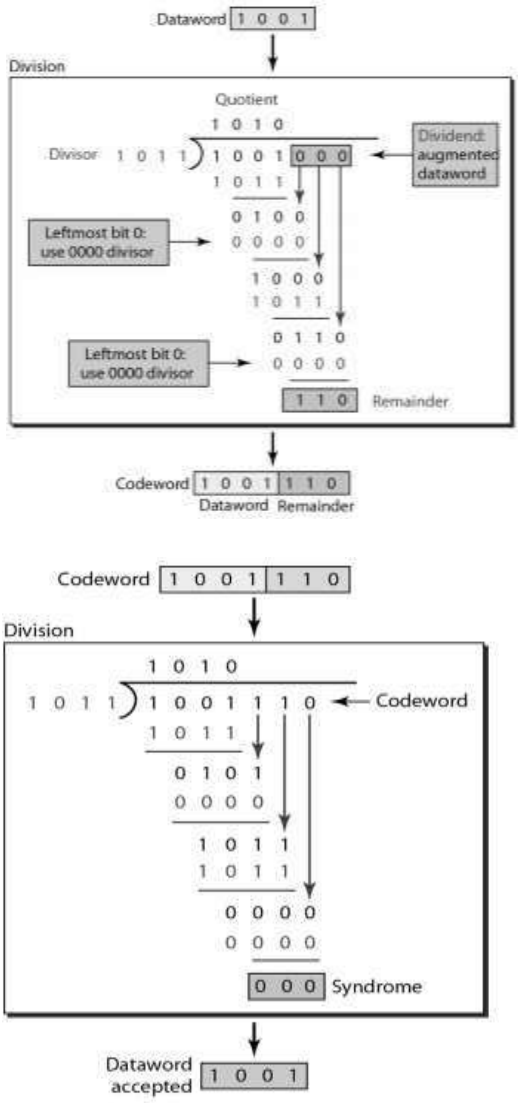

Fig3.3: bit CRC Encoding and decoding

3.4 POLYNOMIAL CODES

• A pattern of Os and 1s can be represented as a polynomial with coefficient of o and 1.

• Here, the power of each term shows the position of the bit and the coefficient shows the values of the bit.

For example, if binary pattern is 100101, its corresponding polynomial representation is $x^5 + x^2 + 1$. Figure shows the polynomial where all the terms with zero coefficient are removed and x J is replaced by x and XO by 1.

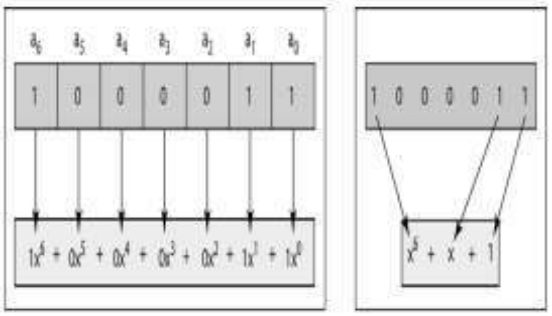

Fig3.4 Binary pattern and its Polynomial representation

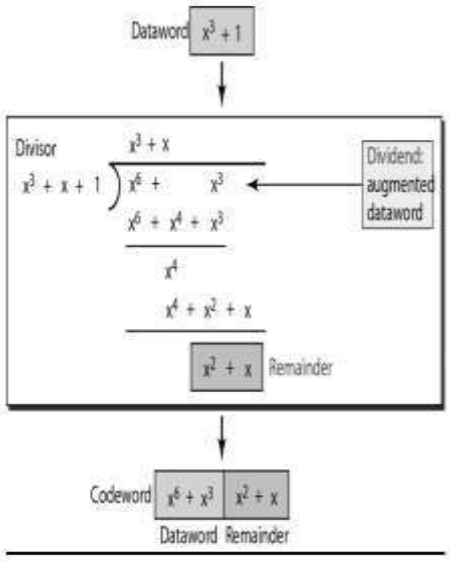

Fig3.5: CRC Division by polynomial

IV. PROPOSED PARALLEL CRC

Cyclic redundancy codes

Cyclic redundancy codes (also known sometimes as cyclic redundancy checks) have a long history of use for error detection in computing among the commonly cited standard reference works for CRCs. A treatment more accessible to non-specialists can be found. A CRC can be thought of as a (non-secure) digest function for a data word that can be used to detect data corruption. Mathematically, a CRC can be described as treating a binary data word as a polynomial over GF (2) (i.e., with each polynomial coefficient being zero or one) and performing polynomial division by a generator polynomial G(x). The generator polynomial will be called a CRC polynomial for short. (CRC polynomials are also known as feedback polynomials, in reference to the feedback taps of hardware-based shift register implementations.) The remainder of that division operation provides an error detection value that is sent as a Frame Check Sequence (FCS) within a network message or stored as a data integrity check. Whether implemented in hardware or software, the CRC computation takes the form of a bitwise convolution of a data word against a binary version of the CRC polynomial.

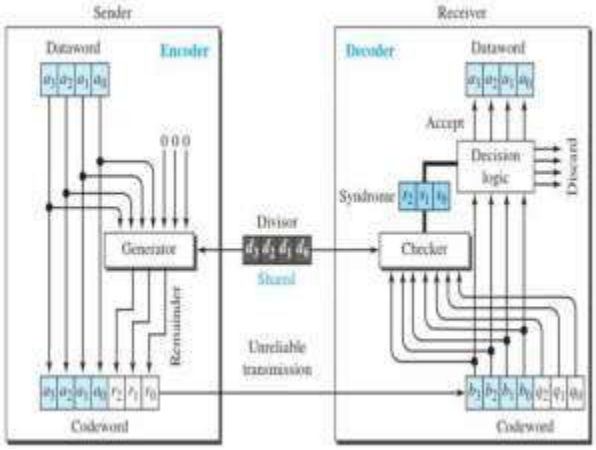

Fig4.1: CRC Encoder and Decoder with sample 4 bit processin

Error detection is performed by comparing an FCS computed on a piece of retrieved or received data against the FCS value originally computed and either sent or stored with the original data. An error is declared to have occurred if the stored FCS and computed FCS values are not equal. However, as with all digital signature schemes, there is a small, but finite, probability that a data corruption that inverts a sufficient number of bits in just the right pattern will occur and lead to an undetectable error.

The minimum number of bit inversions required to achieve such undetected errors (i.e., the HD value) is a central issue in the design of CRC polynomials. The essence of implementing a good CRC-based error detection scheme is picking the right polynomial. The prime factorization of the generator polynomial brings with it certain potential characteristics, and in particular gives a tradeoff between maximum numbers of possible detected errors vs. data word length for which the polynomial is effective. Many polynomials are good for short words but poor at long words, and the converse. There are relatively few polynomials that are excellent for medium-length data words while still being good for relatively long data words. Unfortunately, prime factorization of a polynomial is not sufficient to determine the achieved HD value for any particular message length. A polynomial with a promising factorization might be vulnerable to some combination of bit errors, even for short message lengths. Thus, factorization characteristics suggest potential capabilities, but specific evaluation is required of any polynomial before it is suitable for use in a CRC function. While many previous results for CRC effectiveness have been published, no previous work has attempted to achieve complete screening of all possible 32-bit polynomials.

PROPOSED 32 Bit Parallel CRC ALGORITHM

In order to achieve high speed communication, a parallelarrangement of serial CRC is used. In this the message isdivided into blocks of length k/m bits each. When comparingserial CRC, a parallel CRC can calculate the CRC in k/m clockcycles, and hence throughput can be increased.

However, as with all digital signature schemes, there is a small, but finite, probability that a data corruption that inverts a sufficient number of bits in just the right pattern will occur and lead to an undetectable error.

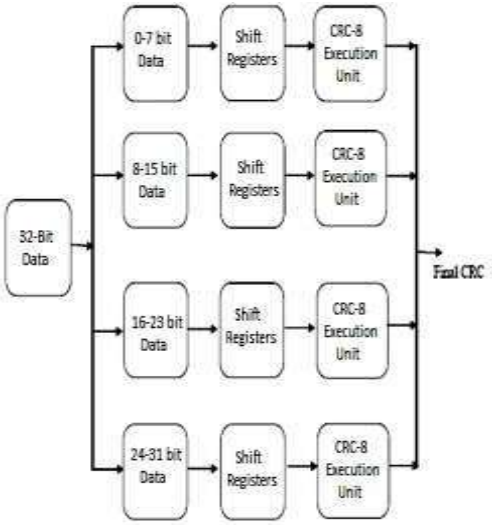

Fig4.2: Parallel CRC

In Fig. above, a parallel CRC structure is shown which receives 32 bit message (k=8), which is divided four blocks (m=4).For processing the message four execution units are there,into which each message block is given. After 8 clockcycles (k/m=32/4=8) four 8 bit CRC values (if degree nof generator polynomial is 8) will be obtained. The final 8 bit CRC will be the xor of these four CRC values. Parallelprocessing increases the throughput rate as well as the no ofbits that can be processed at a time. The proposed architecturecauses less critical path and hence circuit speed get increase. The hardware cost also reduced, which introducesa less complexity in the structure.

V. SIMULATION RESULTS

5.1 EXISTED SYNTHESIS AND SIMULATION RESULTS

Existing 8 bit CRC

Fig 5.1:. Area report of the Existing 8 bit CRC

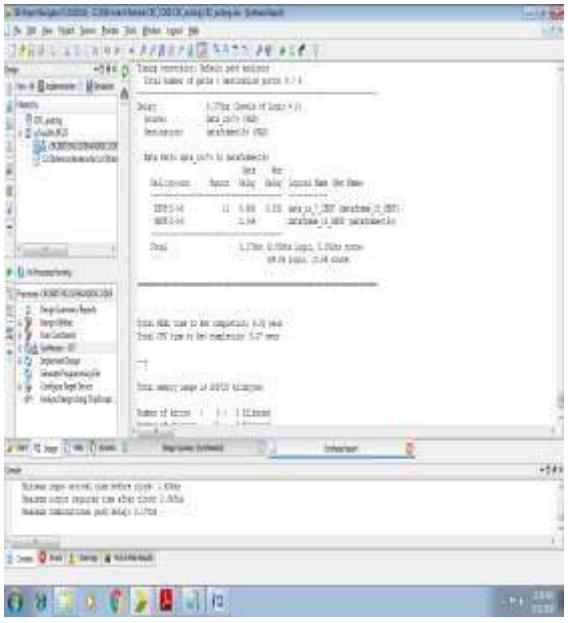

Fig 5.2:Delay report of the Existing 8 bit CRC

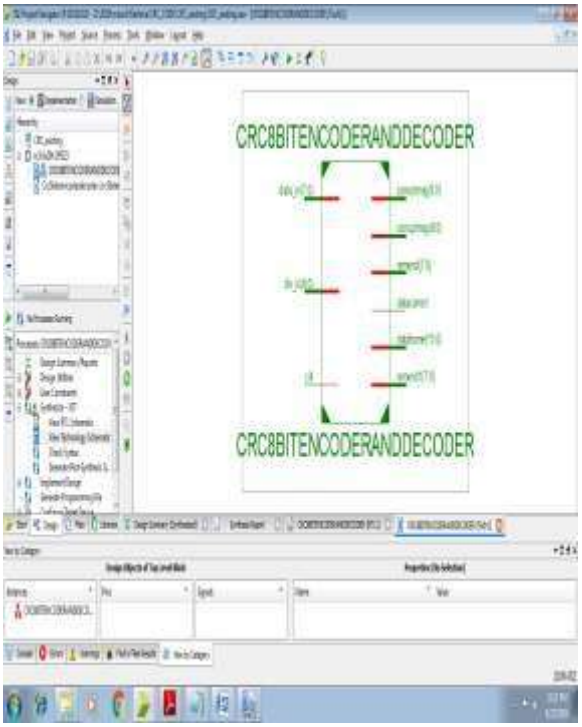

Fig 5.3: Block Diagram of the Existing 8 bit CRC

Fig 5.4:RTL Schematic of the Existing 8 bit CRC

Register Transfer level schematic diagram of the CRC-8 is shown in the above figure which can shows the internal Architecture of the CRC-8

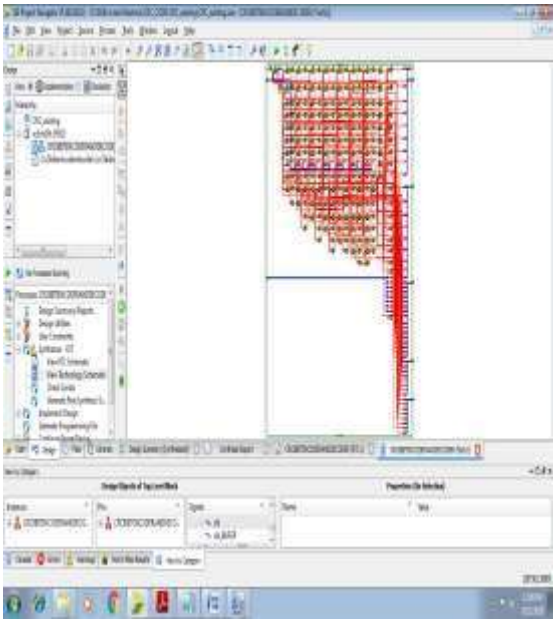

Fig 5.5: Technology Schematic of the Existing 8 bit CRC

Technology Schematic is shown in the above figure which can give us the area details of the CRC-8 where we it consumes 102 Luts for the desin

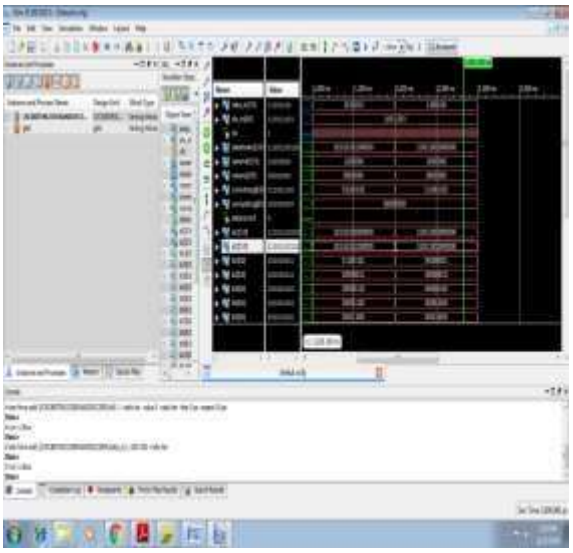

Fig 5.6: Simulation result of the Existing 8 bit CRC

Technology Schematic is shown in the above figure which can give us the area details of the CRC-32 where we it consumes 102 Luts for the each CRC-8 design and we 4 CRC-8 in

tatal sowe have total of 408 and total of 412 LUTs are used to Complete CRC-32

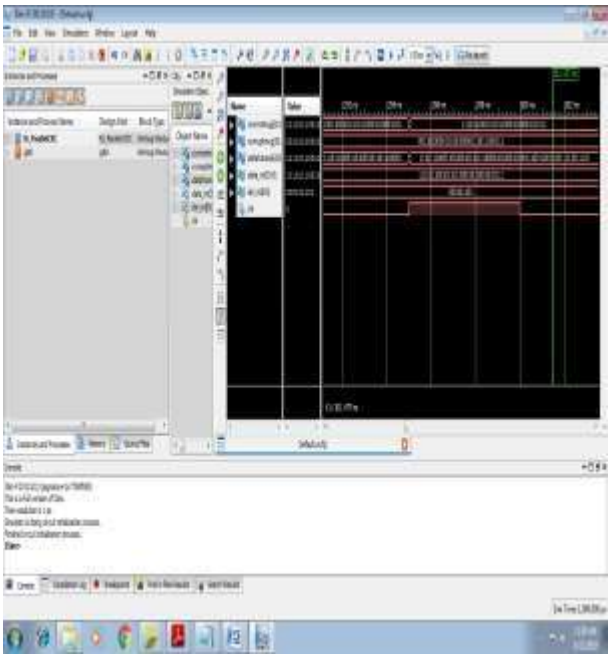

Fig 5.7: Simulation result of the Proposed 32-bit Parallel CRC

Comparision Table:

Parameters	Existing CRC(8bit)	Proposed Parallel CRC(32bit)
Area	102 LUTs	412 LUTs
Delay	3.170ns	3.170ns

Here in this project we had designed the 32 bit Parallel CRC which is designed with parallel architecture of CRC-8 thus can have same delay of CRC-8 for CRC-32 Also but obviously Area Get increased.

CONCLUSION

In this Paper, an High speed parallel CRC architecture is implemented using state space transformation. The main contribution of this paper is implementing CRC-32 in Parallel Manner with the help for CRC-8 thus we have better results in terms of delay and design complexity. Since this method can reduce the circuit complexity, thus speed gets limited and is extended to multi bits data processing. But the power consumption of the proposed architecture increases when compared to the existing architectures, in future, designs to decrease the power consumption along with the area reduction can be included. Thus an area efficient and power efficient parallel CRC encoder can be achieved.

FUTURE SCOPE

The basic work done till now in lte advance is quite useful but number of block still missing which could be implemented further . following things should be done so that our system is more efficiently work :-

1. Forward error correction code will be implemented both tail biting convolution and turbo code as specified in thesis so that it could efficiently transmit data.

2. Automatic Gain Control (AGC) at receiver RF front end – For getting better SNR performance, and Signal to Quantization Noise Ratio (SQNR) at different points of system .

3. Scrambler module and windowing techniques with OFDM generation – To reduce Peak to Average Power Ratio (PAPR) limitation.

4. The whole design may be made more configurable from upper layer like bandwidth selection MIMO mode selection

REFERENCES

[1] O. Shacham, O. Azizi, M. Wachs, W. Qadeer, Z. Asgar, K. Kelley, J. P. Stevenson, S. Richardson,M. Horowitz, B. Lee, A. Solomatnikov, A. Firoozshahian, Rethinking Digital Design: Why Design Must Change [J]. IEEE Micro, 33(6), 2010, 9-24

[2] M. Awad, FPGA supercomputing platforms: A survey, International Conference on Field Programmable Logic and Ap-plications, 2009, 564-568

[3] P. Prasad, C. R. Subrahmanya, A high speed networked signal processing platform for multi-element radio telescopes [J]. Experimental Astronomy, 31(1), 2011, 1-22

[4] Nachiket Kapre, Bibin Chandrashekaranl, Harnhua Ng, Kirvy Teo,Driving Timing Convergence of FPGA Designs through Machine Learning and Cloud Computing,Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on,2015,119-226.

[5] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, R. Laufer, PipeRench: a coprocessor for streaming multimedia acceleration, Proceedings of the 26th International Symposium on Computer Architecture, 1999, 28-39

[6] H. H. Chun, W. Y. Chi, P. Leong, W. Luk, S. J. E. Wilton, Floating-Point FPGA: Architecture and Modeling [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(12), 2009, 1709-1718

[7] C. LaFrieda, B. Hill, R. Manohar, An Asynchronous FPGA with Two-Phase Enable-Scaled Routing, IEEE Symposium on Asynchronous Circuits and Systems (ASYNC), 2010, 141-150

[8] Shanshan Yong,Xin'an Wang, Zheng Xie, Ying Cao, Reconfigurable Operators: New Configuration Logic Blocks for Novel FPGA [J], Journal of Information and Computational Science, 2014, 11(12).

[9] Forouzan, A. B., Data Communications & Networking(sie). Tata McGraw-Hill Education, chapter 10, p.p 265-278, 2011