International Journal of Engineering Innovations in Advanced Technology
ISSN: 2582-1431 (Online), Volume-6 Issue-3, September 2024

Improvement Of Memory Data Corrections By Using CRC Technique For Fault Torrent Applications

Maraboyina Kalyani[1] , Dr S Kishore Reddy[2] , S Sagar[3]

[1]M.Tech Student, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., Hyderabad, India.
[2]Associate Professor, HOD, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., India.
[3]Assistant Professor, ECE, VLSI System Design, Avanthi Institute of Engg. & Tech., Hyderabad, India.

Abstract - A Bose-Chaudhuri-Hocquenghem (BCH) code **decoder that is both highly efficient and consumes very little power when decoding is shown in this paper. DEC-TED is an abbreviation that stands for "double-error-correcting and triple error-detecting." It is an excellent decoder for use in constructing memory to repair mistakes since it is both very efficient and very power efficient. The purpose of our proposal is to develop an adaptive error correcting strategy with the intention of making the process of decoding the DEC-TED BCH code more effective. Immediately after the establishment of a syndrome, this method counts the amount of errors that are present in a codeword and then applies a different error correction algorithm based on the specific error conditions. By increasing the efficiency of the decoding process, the adaptive error correction approach brings about a considerable reduction in both the power consumption and the average decoding delay. further reduce power consumption, we suggest a method called invalid-transition-inhibition. This approach eliminates invalid transitions that are brought about by syndrome vector errors in the error-finding block. The suggested decoders for the (79, 64, 6) BCH code lower power consumption by approximately 70 percent when compared to the traditional completely parallel decoder that operates at a raw bit-error rate of 104-102. These findings were obtained by synthesis utilizing a technology library that is consistent with industry requirements and uses a 65-nm technology.**

Keywords – Memory, CRC, BCH Code,65-nm, Decoding

1. INTRODUCTION

Cyclic redundancy checks, which are methods for error detection, are often used by digital networks and storage devices. The brief check values of the data blocks that are delivered into these systems are decided by the polynomial division of the contents that is still left behind. Taking corrective action is something that may be done prevent data corruption in the case that the check values do not match after the retrieval process. Cryptographic hash functions, also known as CRFs, are given their name due to the fact that they generate the check value that is associated with the verification of data by using a cyclic process. The end result of this is a check value that is twice as large as the first message, but it does not include any new data. It is possible that the widespread usage of CRCs is due, in part, to the fact that they are readily evaluable according to analytical standards and can be implemented in binary hardware. In addition to this, CRCs are very effective at recognizing frequent mistakes that are caused by transmission channel noise. Due to the fact that functions often have a predetermined length, it is usual practice to get the check value by utilizing the length of the function. When the CRC was initially presented in 1961, W. Wesley Peterson was the one who came up with the idea. During the year 1975, a group of scientists from different countries worked together to design and publish the 32-bit CRC function. This function is used in a number of protocols, one of which being Ethernet. Error-detection methods known as cyclic redundancy checks are often used in digital networks and storage devices these days. After polynomial division, the contents of the data blocks that are being introduced into these systems are subject to brief check values, which are decided by the contents that are left behind.

In addition, each block has its own individual needs. Through the process of constructing and carrying out each block in the top module, the design of the objective core architecture is finished. Specifically, the ETI architecture that was proposed was constructed using VHDL, which was utilized for the code. Each each design block receives its own unique code assignment. Xilinx is a tool for simulation and verification that supports a variety of programming languages and languages, including Verilog, VHDL, Verilog system, and mixed-language design. When it comes to this particular instance, the Model Simulator XST Synthesis tool (XILINX ISE 14.5) is used to finish the whole project altogether. The Xilinx software provides you with the ability to get all three of these characteristics. Before commencing the design of microelectronics, Xilinx is used to do a comprehensive verification of the logic circuit architecture. Through the use of Xilinx's simulator that incorporates delay analysis, it is possible to construct and test intricate logic systems.

Using CRC, we were able to spot defects in a data segment, which allowed us to verify that the data transfers from the

transmitter to the receiver were error-free and caused the least amount of power consumption possible.

2. LITERATURE SURVEY

The sofware implementation of error detecting codes that Feldmeier offered was very quick, in contrast to the other components of the communication system, which were thought to be sluggish. The CRC is an example of a rather complicated error detection method that is particularly susceptible to this. CRC, weighted sum codes (WSC), one's complement checksum, Fletcher (1982) checksum, CXOR checksum, and block parity code are some of the approaches that are used in the process of evaluating the effectiveness of the quick software deployment. CRC, or cyclic redundancy checking, was first developed by N. Matloff [2001] as a method for identifying flaws in digital data, but not for correcting them (it is mostly used in the process of data transmission). When a CRC method is being used, the message that is being sent is often accompanied by a checksum, which is a collection of a certain number of check bits.

In the year 2002, D.G. Mavis revealed modern microcircuits along with certain methodologies that were designed to reduce soft error rates. This technique is used in the process of examining deep submicron microcircuits for the presence of alpha and neutron soft faults. The typical static latch SEU (single event upsets) and SET (single event transient)-induced mistakes are both handled by this strengthening approach, which is referred to as temporal sampling.

Through the use of percolation theory, O. Douses and colleagues [2003] conducted an investigation on the influence that interference has on the connectivity of large-scale ad hoc networks. When the signal-to-noise ratio at the receiver is higher above a certain threshold, it is possible for two nodes to create a connection that communicates in both directions. When the network reaches a particular node spatial density, it comprises a vast cluster of nodes that are potentially unlimited. This cluster enables nodes that are located at great distances to interact with each other via a series of hops.

In the article that they published in 2003, Philip Levis and his colleagues proposed modeling the whole Tiny OS application. Utilizing the sensor network domain and the architecture of Tiny OS, TOSSIM makes use of a probabilistic bit error model for the network record network activity with a high degree of accuracy, even when extending to thousands of nodes. The vulnerabilities that were present in Tiny OS ranged from overflows in the queues of the routing protocol to interactions at the bit level that included Message Authentication Codes (MACs).

Within the scope of his study from 2004, M. Chaing suggested that multi-chip transmissions and interference-limited connection rates have to be taken into consideration. maintain architectural flexibility while simultaneously enhancing overall network performance, it is necessary to strike a balance between the management of power at the physical layer and the control of congestion at the transport layer. Distributive power management algorithms, when used in conjunction with typical TCP protocols, have the potential to enhance the efficiency and performance of the end-to-end network while simultaneously reducing the amount of power that is used. This linked technique, when used within the rigorous framework of nonlinearly restricted utility maximization, converges to the global optimum for both synchronous and asynchronous implementations. This makes it possible for geometric convergence to occur while still preserving the modularity that was intended for the transport and physical layers. Through the use of an FPGA in Hukla, he was able to resolve the issue of a single-bit error in the CRC-16 [2004].

Through the use of cyclic redundancy checks (CRCs), framing approaches have the ability to identify transmission faults. CRC is used to encrypt a whole page in the majority of instances; on the other hand, retransmission is required in the event that a mistake is discovered. On the other hand, the capability for correcting errors consisting of a single bit is only available in particular protocols and must be supplied in the header of the frame. It is probable that this has a significant effect on the synchronization function of the receiver. There is a possibility that hardware header error correction will be a problem at 10 Gbps.
R. Henate [2004] developed a method that was revolutionary in that it calculated the amounts of interference that occur in wireless multi-hop ad hoc communities. All of these characteristics are taken into consideration when calculating the carrier-to-interference ratio, often known as C/I. The density of nodes, the number of nodes in the network, the volume of relay traffic, the characteristics of the network's multi-hop operation, and radio propagation variables are some of the elements that are considered. By making use of the anticipated C/I ratios, it is able to make a prediction for the data throughput per node. An authentication methodology that can filter fraudulent data packets that have been injected by a specific number of compromised colluding nodes is provided by S. Zhu et al. [2004] in the context of an interleaved hop-by-hop authentication mechanism. Both the base station and the nodes that are located along the route have a high likelihood of identifying inaccurate data.

3. CRC AND ITS WORKING

The idea of cyclic error-correcting codes is the conceptual foundation upon which CRCs are built. W. Wesley Peterson was the first person to propose the use of systematic cyclic codes as a device for the identification of faults in communication networks in the year 1961. The use of these codes is necessary encrypt conversations. Each one is made up of a check value that is of a certain length. (1) [1] Not only are cyclic codes simple to construct, but they are also excellent at identifying burst errors, which occur when the

incorrect data symbols appear in messages several times. Cyclic coding is a technique that is often used in a variety of computer programming languages. The use of cyclic codes is responsible for bringing about both of these additional benefits. It is essential to avoid making these errors since they have the potential to occur across a wide variety of communication channels, including magnetic and optical storage systems. Despite the fact that an n-bit CRC may be able to detect a single n-bit or shorter error burst, it is also capable of identifying a portion of all bigger error bursts when it is applied to data blocks of any size possible. How come? mainly due to the fact that an n-bit CRC has the capability of detecting an error burst that is n bits more severe than a 1-bit CRC.

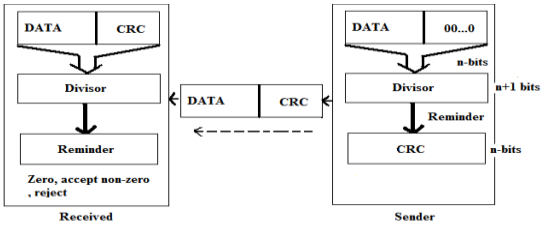

Fig 3.1: Both the transmitter and the receiver are involved in the CRC operation.

The employment of cyclic redundancy check is a typical approach that is used in the area of data transmission as well as related disciplines such as storage and compression for the purpose of addressing data mistakes. Real-time CRC calculation is often accomplished via the use of linear feedback shift registers (LFSRs), which are devices that handle serial data. It is important to note that the throughput of the serial CRC code computation is quite low. Nevertheless, by using concurrent CRC calculations, it is feasible to considerably boost the pace at which computing has been accomplished. Despite the fact that CRC-32's 32-bit parallel computing may be capable of achieving a great deal of gigabits per second, it is not yet a enough performance for applications that need a great deal of speed, such as Ethernet networks. Investigating the CRC-CCITT variants that are used by disk storage, SDLC, XMODEM, the X-25 protocol, and other protocols identify errors in data transmission is yet another strategy that might be taken. The fundamental idea behind these codes serves as the foundation for cyclic error-correcting codes. W. Wesley Peterson was the first person to suggest the use of systematic cyclic codes, which are a way of encrypting communications with a fixed-length check value. This approach was first presented to assist in the identification of communication network failures. Binary polynomial division is the method that is used to construct the CRC, which stands for the Cyclic Redundancy Check. This error-detection code is often utilized. The sender must first transform the binary data into a binary polynomial and then divide the resulting number by a standard generator get a CRC (such as CRC-32) from the binary data. The receiver is given the original data along with the CRC of the data, which is the fraction of this split that is still left behind. Following the receipt of the data and the CRC checksum, the receiver divides the data by two by using the same generating polynomial on both sides of the data. Using the CRC approach, which merely adds 32 bits (in the case of CRC-32) to the message, it is possible to immediately identify a single mistake or a burst of errors. This is true regardless of the size of the data that was initially stored.

A. *SERIAL CRC*

When it comes to the generation of serial CRCs, the usage of linear feedback shift registers (LFSR) has been the norm for a very long time. The only divisions that LFSR can do for CRC computations are binary divisions. When doing binary divisions, shifts and subtractions are often used as the methods of choice. The CRC check is performed on an individual basis here. Each clock pulse will have a single binary value as the data input for that particular pulse. Following the transmission of the 25-bit input message by serial transmission, the CRC-32 algorithm, which is a generator polynomial with one execution unit, is ultimately used to determine the result. Calculating CRC will need more time to complete. A limited bandwidth and a substantial amount of delay are the effects of this. Therefore, solve this issue, we are using a number of devices that do parallel processing.

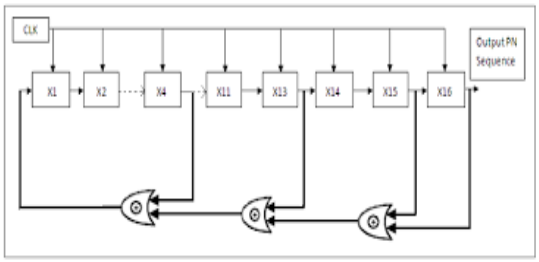

Fig 3.2: Basic LFSR architecture.

LINEAR Feedback Shift Registers, often known as LFSRs, are frequently used in BCH encoders and CRC algorithms. An LFSR circuit that operates sequentially will not be sufficient in circumstances when very high data transmission rates are needed. Due to this constraint, parallel architectures are required to be implemented in applications that need throughputs of several gigabits per second. One example of such an application is optical communication systems. LFSRs are used in a number of other ways, including Design for Test (DFT) and Built-In Self-Test (BIST), as stated in reference [4]. While the DFT makes use of LFSRs to compress answers, the BIST makes use of LFSRs. The use of it may result in the generation of pseudorandom binary test sequences. CRC is adopted by almost all communication protocols since it is a dependable mechanism for identifying mistakes that occur during transmission process. Among the codes that are used in communication systems, BCH codes are among the most widely employed. For this reason, early research on LFSR parallel architectures for BCH and CRC encoders has been carried out satisfy the growing need for processing capacity.

There have been a great number of alternative designs suggested for BCH encoders that are capable of operating at high speeds. do this, generator polynomial division and multiplication algorithms serve as the foundation. One of the most important factors to take into account while thinking about LFSR speed is the hefty price tag. While increasing the number of message bits that are processed does not prevent the critical route from getting longer as a result of parallel processing, it does affect the critical path. The parallel LFSR designs that have been published in the literature often result in a greater critical path as the typical conclusion.

B. P ARALLEL CRC

One method that is used for pipelined CRC calculation is a table-based technique.
You may generate CRCs in parallel using the F matrix. This function refreshes the CRC in a flash.
Fourth iteration of the Re-Tipping and Pipelining Algorithm Throughput may be boosted by using parallel processing, which provides several outputs simultaneously. This allows for greater throughput. Accelerate the clock rate of an integrated circuit by reducing the amount of time required for the computation of the important path. In a rapid CRC update approach, it is sufficient to calculate CRC for the data bits that have been updated; it is unnecessary to compute CRC for all of the data bits. The construction of a parallel CRC may be accomplished in a number of different ways, and each of these methods has both positives and negatives associated with it. On account of the fact that generalized CRC calls for a pre-calculated LUT and rapid CRC update calls for a buffer to retain the previous CRC and data, a table-based architecture is not suitable for this purpose. A greater number of iterations will be required as the design process moves further. Designs that are based on F matrices, which are simple and have a low level of complexity, are becoming more and more prevalent.

Using the parallel LFSR architecture that has been presented, the procedure may be completed more quickly. By reducing the number of steps that make up the main route, pipelining is a technique that may simplify and speed up a process. The design of the LFSR is recomposed here. We will assume, for the purpose of this discussion, that the LFSR's input is u(n) and that its needed output is y. (n).

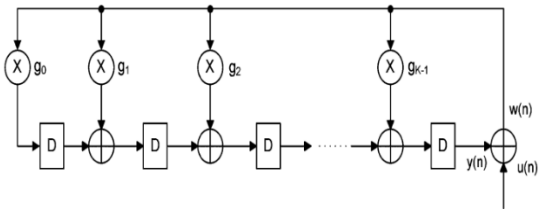

Fig 3.3: Standard LFSR designs

Through the use of this design, it is possible to get four bits of data input for each clock pulse by increasing the size of the data input and lowering the clock pulse. It will reduce the number of clock pulses that are required, which will allow for the task to be completed in a timely and effective manner. There are four 64-bit blocks, and each of them has 64 bits. The data is stored in each of these blocks. Their range of values is from one to sixteen. In this particular case, there are four execution units being used. It is possible to construct four 32-bit remainders due to the fact that they are pipelined in concurrently. The characteristics that they possess include an ex-or result and a 32-bit final CRC.

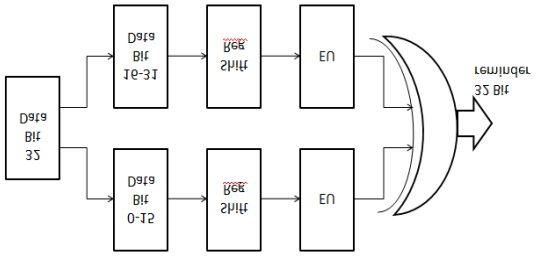

Fig 3.4: For the purpose of CRC computation, this highlights the use of four parallel execution units

You may also generate a checksum by adding up all of the bytes in the message, or you can use the exclusive or function. Both of these methods are feasible alternatives. The latter technique includes adding the carry from each 8-bit total to the bit of the accumulator that is the least significant. When compared to the basic exclusive or the total of the bytes with carry deleted, this is more likely to detect faults, according to the information provided by a number of specialists. One method for detecting errors is the cyclic redundancy check, which has received a lot of praise and is simple to apply in hardware development. Within the context of this particular case, a checksum that is either eight, sixteen, or thirty-two bits in length is attached to the message. In this section, we will first go over the foundations of 32-bit CRC checksum computation in software before moving on to various practical ways. Polynomial arithmetic, and more specifically the determination of the remainder of dividing two polynomials in GF (2), which is a Galois field with two elements, is the foundation upon which the CRC is built. To restate it another way, it is the same as if the message were a giant binary integer that could be divided by a large prime number, and the result from that division was determined. The coefficients of the polynomial that corresponds to every given variable x in GF (2) are either zero or one, depending on the variable. Both of these procedures are functionally equal due to the fact that they are both implemented, respectively, in Module 2. There are two sets of polynomials that need be considered: x3 + x+1 and x4 + x3 + x2 + x. In addition to the fact that their sum and difference are identical to one another, their total is also identical to one another. This is due to the fact that a coefficient of -1 is comparable to a coefficient of 1, which is the reason why negative signs are seldom assigned to these polynomials. It is not difficult to multiply polynomials of this kind. If you

multiply two variables by each other, you will get the same result as if you added them together using logical and operation. You may use exclusive or to sum up the intermediate results you obtain from multiplying the variables by each other. Performing the calculation of the CRC checksum does not need the use of multiplication any more. do long division of polynomials over integers, a similar method may be used to divide over GF (2).

4. METHODOLOGY

A. Error detection scheme in BCH

Hash functions are often used for error detection since they operate as checksum algorithms. As a result of recalculating the tag and comparing it to the one that was provided, hash functions make it possible for receivers to validate the message that was transmitted. There are a great number of different ways that hash functions may be implemented. As an example, the cyclic redundancy check has become a widely used technique for identifying burst faults as a result of its capacity to identify defects of this kind.

Cyclic redundancy checks, often known as CRCs, are a kind of hash function that is not secure. They are used by computer networks discover any unintended alterations that may have been made to digital data. However, CRCs are unable to identify vulnerabilities that have been maliciously created. Once a generating polynomial is provided, the data that is provided as input is used as the dividend in a polynomial long division, and the result is the remainder of the division. This is the definition that will be used.

One of the many advantageous characteristics of cyclic coding is its fast identification of burst faults, which is only one of its many advantages. It is common practice to make use of CRCs in digital networks and storage devices like hard disk drives because of the simplicity with which they may be implemented in hardware. One specific use of CRC is cyclic redundancy check (CRC) parity, which generates a single-bit CRC by using the divisor x plus 1.

a. Error correction in BCH

As a method for managing errors that occur during data transmission, ARQ makes use of error detection codes, acknowledgment and/or negative acknowledgement messages, and timeouts. This is done ensure that data delivery is reliable. In the event that an acknowledgment message is sent by the receiver, the data frame is regarded as having been successfully received by the device that is receiving it.

Generally speaking, the transmitter will retransmit the data frame after it has been delivered for a certain period of time. This will continue until the data frame is either successfully received or the error continues to be present after a predetermined number of retransmissions. The approaches of Stop-and-wait, Go-Back-N, and Selective Repeat are all examples of ARQ procedures.

B. Algorithm of BCH

Polynomial division, modulo two, is used to compute cyclic redundancy checks. The "generator polynomial" string divides the binary message string by a set number of zeroes, but instead of subtractions, exclusive OR operations are used. Hardware-based implementations of this type of division use a modified shift register[1], while software implementations use a series of equivalent algorithms that begin with simple code close to the arithmetic before becoming faster (and, in some cases, more obfuscated[2]) as a result of byte-wise parallelism and space-time trade-offs.

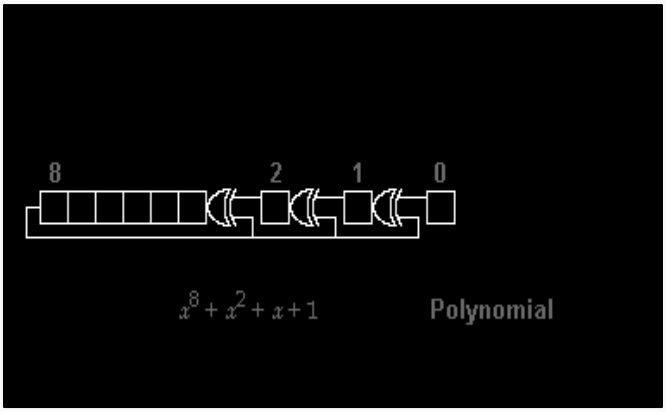

Fig 4.1: polynomial equation

An example of how to generate an 8-bit CRC is shown below. The generator is a shift register of the Galois type with xor gates arranged in accordance with the generator polynomial's powers (white numbers). Any length is acceptable for the message stream. The result in the registration is the checksum after it has been moved through the register and followed by 8 zeroes.

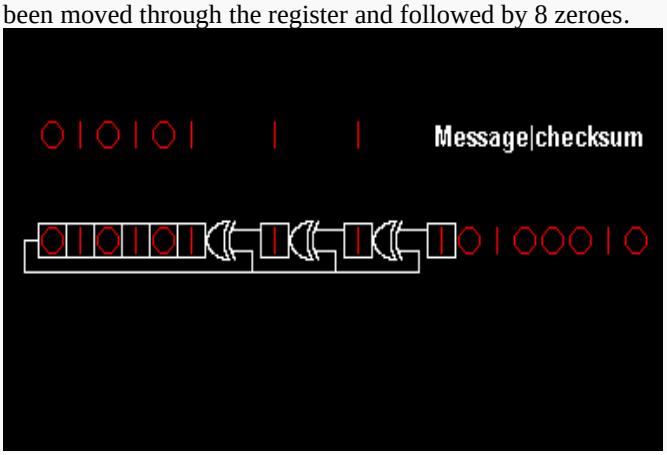

Fig 4.2: generating 8-bit BCH

When receiving data, verifying the checksum. Checksum received instead of zeroes is appended to received message when it is shifted via same register as generator. While a message or checksum with a single faulty bit will provide an all-zero result, the presence of a different result indicates an issue.

Specific shift register values and a final 'm Available step are added to the polynomial division process by various CRC standards (bendiness). Code in practise may stray from "pure" division,[2] resulting in a shifted or shifted "register."[3]

An n-bit CRC has been generated using this pseudo code. You can't just use an integer when dealing with polynomials; you have to create an object that could be added to, multiplied by, and exponented. When two polynomials are XORed, the values of each matched term from both polynomial are exclusive ORed.

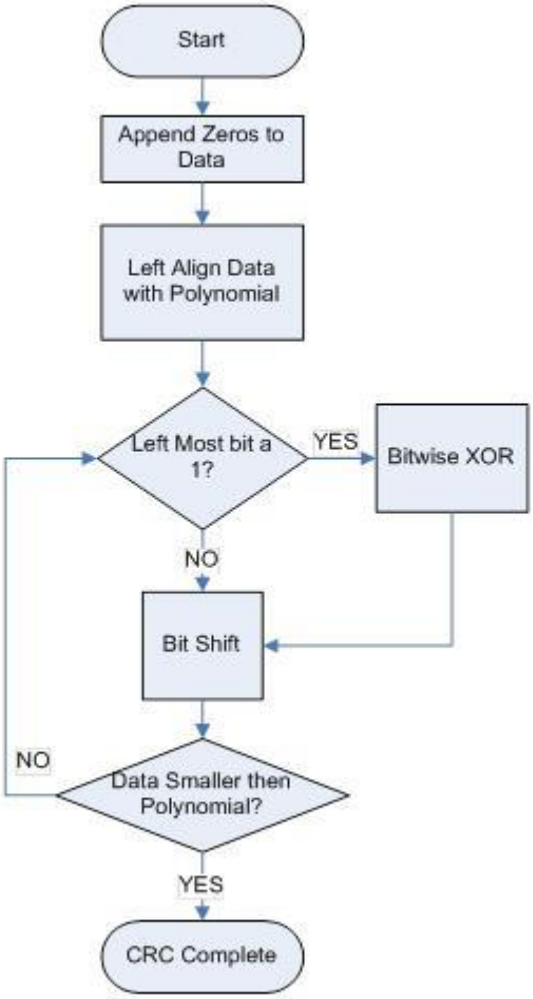

Fig 4.3: algorithm of CRC

The majority of standards adhere to the standard procedure, which consists of performing a CRC reversal after setting the register to all ones. Due to the fact that the CRC is able to recognize any extra bits that are added to the message, the message's integrity and any modifications that are made to it are not harmed.

On the basis of whether or not the messages are multiples of the CRC polynomial, the fundamental mathematics of a CRC decides which messages are permitted (considered to have been successfully transmitted) and which ones are refused. There is still the possibility of making this message seem to be a polynomial by adding a few more leading zero bits at the beginning of the message. One may claim that the number 1 and the number 000001 are indistinguishable from one another.

The standard CRC technique is unable to determine whether or not the number of leading zeros has changed in this specific circumstance since it is unable to determine either of these things. It is possible for these bits to be added to the stream of data accidentally during the process of data transmission if the rem shift register is originally set to a value that is not zero, such as "all ones." Based on the assumption that the CRC register contains a total of n bits, this notion is comparable to the process of mathematically augmenting the first n bits of a message.

If the initial values that are utilized by the generator and the checker are the same, then the creation and checking of CRCs will not be affected in any way. Any number that is not zero may be used as a beginning point; however, the all-ones value, also known as the 1 in twos complement binary, is the most popular starting point. A limited set of specific criteria serve as the foundation for specialized principles. If the message is incorrect, a one-pass CRC check or generate will still provide a result of 0. This is because the message is only checked once.

It is possible that the same sort of problem will occur when a communication is being closed off. Taking into consideration that the polynomial of the message is already a multiple of the CRC polynomial, the outcome will not change with this assumption. This is equivalent to adding x to the polynomial that represents the message. As is the case with 726, this is also the case with 7260, which is a multiple of 11 in the same way. Before the CRC register is included in the message, it is possible to invert it in a manner that is analogous to the one that may be employed while the message is being processed after it has been received. A pattern consisting of all ones, which is then XORed with a pattern consisting of all zeros, is not necessary modify any number of bits.

5. EXPERIMENTAL RESULTS

A. SIMULATION

VHDL was used to calculate the important area delay of the design, and Xilinx ISE 145i was applied for simulation. ModelSim was utilized for the simulation. Due to the larger circuit size that is needed by the improved ETI design with an XOR gate, there is a delay in the implementation process as a consequence of this necessity. Guarantee that your design will function as intended, you should run it through a simulation. After you have completed your design and coding, the next step is to go to this phase. You are able to put your idea to the test by using ModelSim and other simulators that are comparable. There is another name for this technique, which is functional simulation. In other words, simulation is nothing more than a means for determining whether or not a piece of hardware has the desired logical capabilities, but it does so without taking into account real temporal concerns such as delays in the network or circuits.

B. SYNTHESIS

For the purpose of generating a gate-level net list, the process of synthesis requires the use of a VHDL model of a circuit at the register-transfer level. At the register transfer level, blocks such as multiplexers and arithmetic logic units are linked to one another by wires during the synthesis phase of the net list generation process. This phase is responsible for the formation of a net list. By "synthesis," we imply that the process of transforming a concept into physical hardware is what we mean when we speak about it. If you are looking for validation of your functional design on a level that goes beyond the cerebral, then Synthesis is the way to go. When we have finished validating your design, we will go on to the next step, which is the implementation of the hardware. accommodate this, it is necessary to shift away from RTL and toward gate level architecture.

The three phases that are required for synthesis are as follows: In addition to mapping technology, translating and optimizing content is also supported. At the gate level, the second need is to translate RTL into net-lists. There is also optimization-technological logic, which is not optimization. This brings us to the third point. When optimization is performed, the number of components that are necessary to provide the desired functionality is decreased. In addition, there is a "timing simulation" that may be used.

C. EXPERIMENTAL RESULTS

Simulation Results for 32 bit CRC:

The 32 bit CRC with XOR gate is coded in VHDL and the design is simulated using and Xilinx ISE 14.5i, Spartan 3E.

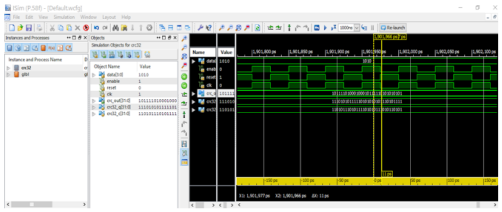

Fig 5.1: Simulation results of 32 bit CRC

Fig 5.1: Simulation Results Of 32 Bit CRC

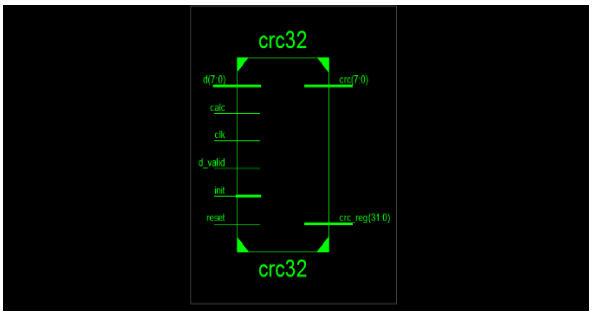

Fig 5.2: RTL schematic diagram for CRC

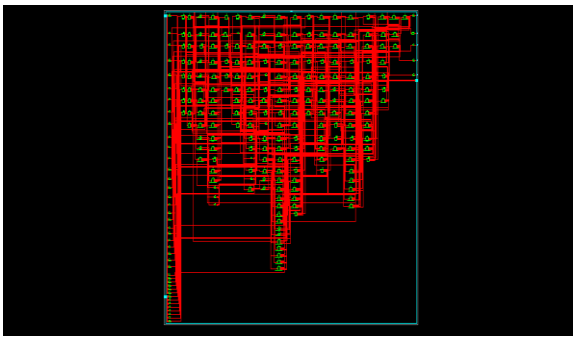

Fig5.3: internal schematic diagram for CRC

CONCLUSION AND FUTURE SCOPE

CONCLUSION

However, because to its limited throughput, serial technology is often not used when it is necessary to transport huge volumes of data in a short length of time. Parallel implementation is the way to go because of how quickly it can be done. When utilizing CRC-32, the transmission of 64 bytes of data takes 17 clock cycles to complete. However, in contrast to CRC-64, the transmission of this data requires nine clock cycles to be completed. In other words, it more than doubles the throughput while simultaneously cutting the processing time in half.

FUTURE SCOPE

To eliminate data mistakes in high-speed DSP communications, CRC will be used primarily in the future, as well as in all

network streams to prevent data mistakes from transmitter to the receiver in all network-related devices.

REFERENCES

[1] Peterson, W. W. And Brown, D. T. (January 1961). "Cyclic Codes For Error Detection". Proceedings Of The Ire 49 (1):228–235. Doi: 10.1109/Jrproc. 1961. 287814.

[2] Ritter, Terry (February 1986). "The Great Crc Mystery". Dr. Dobb's Journal 11 (2): 26–34, 76–83. Retrieved 21 May 2009.

[3] Stigge, Martin; Plötz, Henryk; Müller, Wolf; Redlich, Jens-Peter (May 2006). Reversing Crc – Theory And Practice. Berlin: Humboldt University Berlin. P. 17. Retrieved 4 February 2011."The Presented Methods Offer A Very Easy And Efficient Way To Modify Your Data So That It Will Compute To A Crc You Want Or At Least Know In Compute To A Crc You Want Or At Least Know In Advance."

[4] Cam-Winget, Nancy; Housley, Russ; Wagner,David; Walker, Jesse (May 2003). "Security Flaws In802.11 Data Link Protocols". Communications Of Theacm 46 (5): 35–39. Doi : 10.1145/769800. 769823.

[5] Williams, Ross N. (24 September 1996). "A Painless Guide To Crc Error Detection Algorithms V3.00". Retrieved 5 June 2010.

[6] Wh; Teukolsky, Sa; Vetterling, Wt; Flannery, Bp (2007). "Section 22.4 Cyclic Redundancy And Other Checksums". Numerical Recipes: The Art Of Scientific Computing (3rd Ed.). New York: Cambridge University Press.Isbn 978-0-521-88068-8.

[7] Koopman, Philip; Chakravarty, Tridib (June 2004)."Cyclic Redundancy Code (Crc) Polynomial Selection For Embedded Networks". The International Conference On Dependable Systems And Networks: 145 – 154. Doi : 10.1109/Dsn. 2004.1311885. Isbn 0-7695-2052-9. Retrieved 14 January 2011.

[8] Cook, Greg (6 July 2012). "Catalogue Of Parametrised Crc Algorithms". Retrieved 7 July 2012.

[9] T.-B. Pei, And C. Zukowski, "High-Speed Parallel Crc Circuits In Vlsi,"Ieee Trans. On Commun., Vol. 40, No. 4, Pp. 653-657, Apr. 1992.

[10] J. H. Derby, "High-Speed Crc Computation Using State-Space Transformations," Proc. Ieee Global Commun. Conf., Pp. 166-170, Nov. 2001.

[11] C. Kennedy And A. Reyhani-Masoleh, "High-Speed Crc Computations Using Improved State-Space Transformation," Proc. Ieee Intl. Conf. Electro/Info. Tech., Pp. 9-14, 2009.

[12] G. Hu, J. Sha, And Z. Wang, "High-Speed Parallel Lfsr Architectures Based On Improved State-Space Transformations," Ieee Trans. On Vlsi Syst. Vol. 25, No. 3, Pp. 1159-1163, Mar. 2017.

[13] M. Ayinala And K. K. Parhi, "High-Speed Parallel Architectures For Linear Feedback Shift Registers," Ieee Trans. On Signal Process., Vol. 59, No. 9,Pp. 4459-4469, Sep.2011.

[14] J. Jung, Et. Al., "Efficient Parallel Architecture For Linear Feedback Shift Registers," Ieee Trans. On Circuits And Syst.-Ii, Vol. 62, No. 11, Pp. 1068-1072, Nov. 2015.