

## CHIPSCOPE Implementation of CRC circuit architecture

G.Shanthi<sup>1</sup>, Dr.L.Padmasree<sup>2</sup>

<sup>1</sup>(Lecturer, ECE,VNRVIET/JNTUH, India)

<sup>2</sup>(,Professor, ECE,VNRVIET/JNTUH, India)

---

**Abstract:** *Cyclic Redundancy Check is an established technique for detecting errors on serial data communication links and in mass storage devices. A frame check sequence is appended to the message for transmission error detection in Many (high-speed) serial data communication protocols. The common hardware solution for CRC calculation is the linear feedback shift register (LFSR), consisting a few Flip Flops and Logic Gates. CRC is the preferred and most efficient method used for detecting bit errors produced from medium related noise. Data storage is another area where CRC error detection is becoming increasingly important. CRC checks to be executed at high speed as well as parallel processing. The ability of CRC implemented in hardware to be reconfigurable to handle new Generator polynomials will offer a key advantage in this fast developing area. The reconfigurable CRC circuit that has been implemented can quickly switch between any polynomial gives it a key advantage over the other circuits. In this paper it is proposed **CHIPSCOPE Implementation** of a fully field programmable, architecture for a CRC computation circuit. The resulting architecture is able to support all types and sizes of CRC polynomial, for all types of protocols and data encryption. The architecture has been authored in VHDL, synthesized, and implemented using Xilinx ISE Foundation 10.1i, chipscope. For Behavioral simulation, Place and Route simulation Modesim6.0 is used.*

**Key words :** *CRC (cyclic Redundancy check) ,FEC(Forward Error correction), Generator polynomial, LFSR(Linear feed back Shift Register), Reconfigurable circuit.*

---

### I. Introduction

In information theory and coding theory with applications in computer science and telecommunication, error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, and error correction enables reconstruction of the original data. Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error detection techniques are that enable reliable delivery of digital data over unreliable communication channel. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors.

The general idea 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, and to recover data determined erroneous. The transmitter sends the original data, and attaches a fixed number of check bits or parity data, which are derived from the data bits by some deterministic algorithm. If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits, if the values do not match, an error has occurred at some point during the transmission.

A CRC is one of the an error-detecting code. Its computation resembles a polynomial long division operation in which the quotient is discarded and the remainder becomes the result, with the important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field. The length of the remainder is always less than the length of the divisor called the generator polynomial, which therefore determines how long the result can be. The definition of a particular CRC specifies the divisor to be used, among other things. An important reason for the popularity of CRCs for detecting the accidental alteration of data is their efficiency guarantee. Typically, an n-bit CRC, applied to a data block of arbitrary length, will detect any single error burst not longer than n bits and will detect a fraction  $1-2^{-n}$  of all longer error bursts. Errors in both data transmission channels and magnetic storage media tend to be distributed non-randomly i.e. are 'bursty', making CRCs' properties more useful than alternative schemes such as multiple parity checks. The simplest error-detection system, the parity bit, is in fact a trivial 1-bit CRC, it uses the generator polynomial  $x+1$ .

Cyclic redundancy check (CRC) is an error detecting code that is widely used to detect corruption in blocks of data that have been transmitted or stored. A stand alone intellectual property (IP) core is ideal for accelerating CRC computation in many network and server applications. Hardware configurability that will allow unrestricted CRC sizes and polynomials enables a wide range of network transmission, storage and security applications to be supported at a low cost. The cost of chip design continues to increase due to factors such as high mask and respin costs. Next generation system-on chip (SoC) designs are highly expensive and therefore must be configurable to a range of applications and future proof where either product updates or protocol migration can occur. The fact that the reconfigurable CRC circuit that has been implemented can quickly switch between any polynomial gives it a key advantage over the other circuits referenced, in terms of flexibility and ease of upgrade for new and emerging applications and standards.

## II. Computation of CRC

CRC functions have been widely implemented in software using methods such as lookup tables[4] and addition[5]. CRC is most efficient method used for detecting bit errors produced from medium related noise .CRCs are particularly easy to implement in hardware. Cyclic Redundancy Check (CRC) is an error detecting code that is widely used to detect corruption in blocks of data that have been transmitted or stored. Enables a wide range of network transmission, storage and security applications to be supported at a low cost. CRCs are specifically designed to protect against common types of errors on communication channels., it is a single or burst error detecting code designed to detect accidental changes to digital data in computer networks. It is characterized by specification called G(x), **Generator Polynomial**. Goal to maximize the probability of detecting an error.

To compute an n-bit binary CRC, place the bits representing the input in a row, and position the (n+1)-bit pattern representing the CRC's divisor called a Generator polynomial underneath the left-hand end of the row. The first calculation for computing a 3-bit CRC:

```
11010011101100 <--- input
1011             <--- divisor (4 bits)
-----
01100011101100 <--- result
```

If the input bit above the leftmost divisor bit is 0, do nothing and move the divisor to the right by one bit. If the input bit above the leftmost divisor bit is 1, the divisor is exclusive-ORed into the input (in other words, the input bit above each 1-bit in the divisor is toggled). The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row.

The last calculation is as follows:

```
00000000001110 <--- result of previous step
1011             <--- divisor
-----
0000000000101 <--- remainder (3 bits)
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These n bits are the remainder of the division step, and will also be the value of the CRC function.

## III. Over view of CRC Methodology

### 3.1 CRC Algorithm

CRC is a polynomial based error detecting method. A set of check bits are to be computed for each frame and appended to the end of frame. The CRC value is computed as the remainder of the Modulo-2 division of the message that is to be transmitted and the selected Generator polynomial. From the Message the Message polynomial has to be formed. It consists of n<sup>th</sup> degree polynomial in this the value of each bit is a coefficient

- Example:  $1\ 0\ 0\ 1\ 1\ 1\ 0\ 0$   
 $x^7\ x^6\ x^5\ x^4\ x^3\ x^2\ x^1\ x^0$

$$M(x) = x^7 + x^4 + x^3 + x^2 - \text{Message Polynomial.}$$

The Coefficients with zeros will be vanished. With the remaining terms Message polynomial will be formed.

Step1: Compute  $M(x) * X^k$ , equivalent to adding k No. of zeros.

Example:  $M(x) = 1000$ ,  $G(x)$  of degree 2

$X^3 * X^2 = X^5 = T(x) (10000)$

Step2: Divide  $T(x)$  by  $G(x)$

Step3: Find remainder  $R(x) = T(x) / G(x)$

Step4: Replace the Zeros that are added to the message Polynomial with the remainder forms  $D(x)$ ,  $D(x)$  is exactly divisible by  $G(x)$

Step5: Transmit  $D(x)$

Where  $M(x)$  is the Message polynomial and  $K$ - is the highest order of the Generator polynomial  $G(x)$ . This total product forms the an intermediate polynomial  $T(x)$ . A CRC vale is calculated as a remainder of the modulo-2 division. The remainder  $R(x)$  is appende to the end of the message for transmission that forms the polynomial  $D(x)$ . At the receiver the process of division is carried out. Message appended with remainder divide by the same generator polynomial gives you the remainder.. If there is no remainder i,e remainder is zero no errors are encountered. Fig :1 shows the pictorial representation for the CRC Transmitting frame calculation.

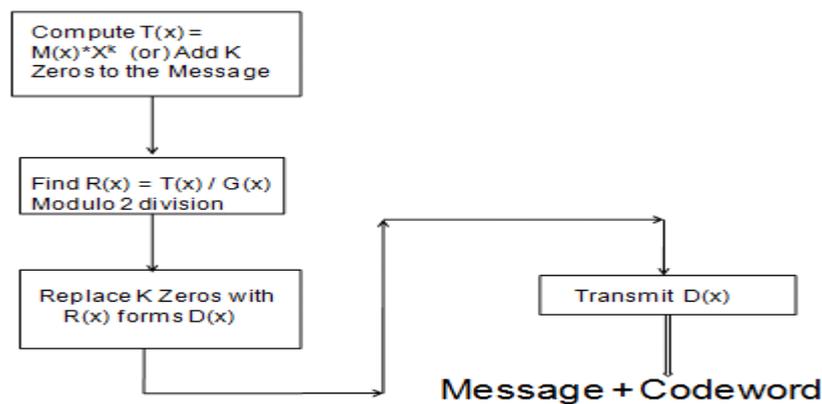


Fig : 1 Flow chart of CRC Method

#### IV. Polynomial selection

##### 4.1 Designing CRC polynomials

The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximise the error detecting capabilities while minimising overall collision probabilities. The most important attribute of the polynomial is its length (the number of the highest nonzero coefficient), because of its direct influence of the length of the computed checksum.

The most commonly used polynomial lengths are:

- 9 bits (CRC-8)
- 17 bits (CRC-16)
- 33 bits (CRC-32)
- 65 bits (CRC-64)

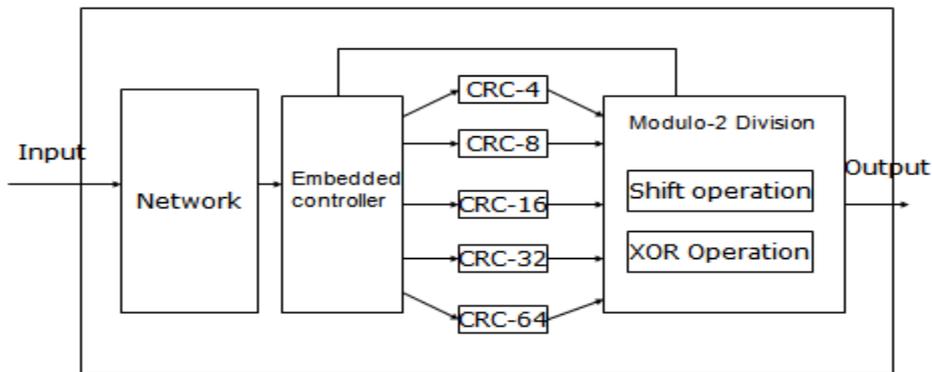
The design of the CRC polynomial depends on what is the maximum total length of the block to be protected (data + CRC bits), the desired error protection features, and the type resources for implementing the CRC as well as the desired performance. A common misconception is that the 'best' CRC polynomials are derived from either an irreducible polynomial or an irreducible polynomial times the factor  $(1 + x)$ , which adds to the code the ability to detect all errors affecting an odd number of bits. In reality, all the factors described above should enter in the selection of the polynomial.

The advantage of choosing say a primitive polynomial as the generator for a CRC code is that the resulting code has maximal total block length; in here if  $r$  is the degree of the primitive generator polynomial then the maximal total block length is equal to  $2^r - 1$ , and the associated code is able to detect any single bit or double errors. If instead, we used as generator polynomial  $g(x) = p(x)(1 + x)$ , where  $p(x)$  is a primitive polynomial of degree  $r - 1$ , then the maximal total block length would be equal to  $2^{r-1} - 1$  but the code would be able to detect single, double, and triple errors. A polynomial  $g(x)$  that admits other factorizations may be chosen then so as to balance the maximal total block length with a desired error detection power. A powerful class of such polynomials, which subsumes the two examples described above, is that of BCH codes. Regardless of the reducibility properties of a generator polynomial of degree  $r$ , assuming that it includes the '+1' term, such

error detection code will be able to detect all error patterns that are confined to a window of  $r$  contiguous bits. These patterns are called ‘error bursts’.

**4.2 Candidate polynomial selection**

The selection of a ‘good’ polynomial for generic use is of course a matter of engineering judgment. The following selection process was chosen to result in polynomials that primarily maintained high Hamming Distance(HD) values to the longest data word lengths possible, secondarily achieved good performance at shorter lengths, and thirdly achieved good performance at longer lengths than the stated maximum usage length. The prioritization of these goals keeps in mind that embedded network applications typically have a maximum message length that needs a certain HD that short messages can benefit from improved HD protection so long as protection of long message is not materially sacrificed, and that sometimes a protocol revision adds messages longer than originally envisioned, so good performance at longer message lengths is desirable as a safety net. Fig. 2 shows the diagram illustrating the selection of generator polynomial depending on the Message bits that are to be transmitted.



**Fig 2 : CRC Architecture**

**V. Implementation of CRC Architectue**

CRC is a well-established method to allow detection of a wide range of data transmission errors. A small number of bits is added to the end of a long message so that the entire bit stream, when treated as a polynomial (or number), is divisible by a particular key Generator polynomial that is programmed into both the sender and receiver. CRC is a polynomial-based block coding method for detecting errors in blocks or frames of data. Number of Zeros equal to the highest order of the message polynomial is to be added to the data. This is equivalent to multiplying  $M(x)$  by  $2^n$ , where  $n$  is the number of FCS digits. After the modulo two division operation we will find the remainder. The computed bits are replaces the number of zeros added in the before operation. A CRC value is calculated as a remainder of the modulo-2 division of the original transmitted data with a specific CRC generator polynomial.

The division uses modulo-2 arithmetic, where each digit is independent of its neighbour and numbers are not carried or borrowed, thus addition and subtraction are performed via an exclusive-OR (XOR) function. The remainder  $R(x)$  is appended to the end of the message before transmission. At the receiver, the message plus the FCS is divided by the same polynomial. If the remainder is zero then it can be assumed that no error has occurred. To implement the 64-bit CRC using the 32-bit polynomial value first 32-bit operation will be performed the value will be stored in the output register again the value will be fed back to the Register which stores the last cycle CRC data value. The input data enters the array down the columns and the outputs are formed along the rows. The current CRC value is held in a register at the array output, which is fed back and XORed with the input data of the next clock cycle as part of the CRC computation process. The outputs are then stored in the registers for the next clock cycle. The computation of the next row depends on the result of the previous row.

The architecture of CRC circuit is composed of different registers that are used to store data at different levels, the configurable XOR gates, Linear feed back shift registers are used to shift the data when Xor operation is performed. The input data enters the architecture from input data Register. The current CRC value is held in a register at the output, which is fed back and XORed with the input data of the next clock cycle as part of the CRC computation process. The outputs are then stored in the registers for the next clock cycle. The desired CRC polynomial and the input port size are stored in registers. where the computation of each row is based on the result from the previous row. Fig 3 shows a diagram illustrating the architecture of the CRC circuit.

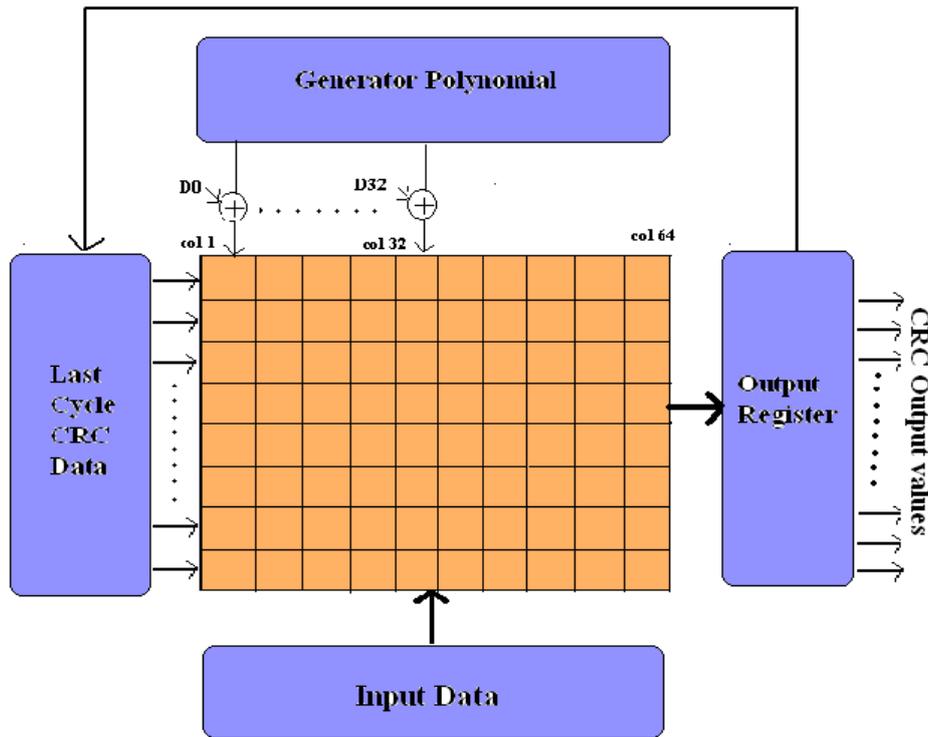


Fig 3 : CRC circuit architecture

### VI. Simulation Results and Performance Analysis

The Architecture shown in figure. 3 has been implemented in VHDL for 64-bit CRC circuit and its Synthesis, Mapping was done with Xilinx ISE 10.1 For Behavioral simulation, Place and Route simulation Modesim6.0 is used. The resulting architecture is able to support all types and sizes of CRC polynomial, for all types of protocols and data encryption. The VHDL Module of 64-bit CRC is shown in fig.4.

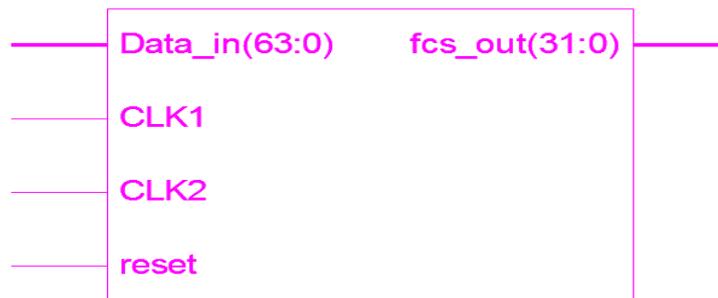


Fig.4 VHDL Module of 64-bit CRC

The Hardware description languages are most used for is the Register Transfer Level (RTL). Between gate level on the low abstraction side and system level on the high abstraction side, The RT level of abstraction is a good balance between corresponds to actual hardware and ease of description for hardware designers. At this level of abstraction designs can be simulated with HDL simulators, they are synthesizable and automatic generation of hardware is provided by most hardware design EDA tools. Routing and Placement phase decides on the placement of cells of the target hardware. Wiring inputs and outputs of these cells through wiring channels and switching areas of the target hardware are determined by the routing and placement phase. The output of this phase is specific to the hardware being used and can be used for programming an FPLD or Manufacturing an ASIC. RTL schematic 64-bit CRC circuit is shown in figure 5. In integrated circuit design, register transfer level (RTL) description is a way of describing the operation of a synchronous digital circuit. In RTL design, a circuit's behavior is defined in terms of the flow of signals or transfer of data between hardware registers, and the logical operations performed on those signals. After the HDL synthesis phase of the synthesis process, use the RTL Viewer to view a schematic representation of the pre-optimized design.

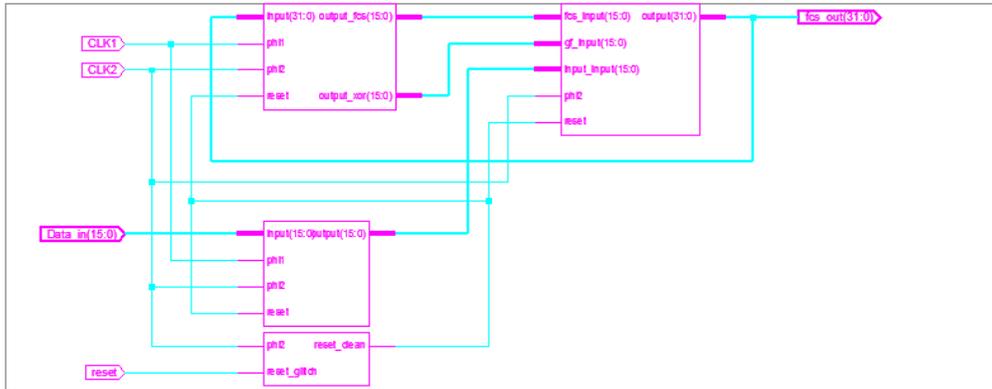


Fig 5 : RTL schematic of 64 -bit CRC circuit generated by the Synthesis

### VII. Simulation Results

The simulation is done by using the Modelsim6.0 simulation and the synthesis is done by using the Xilinx ISE synthesis tool. Behavioral simulation waveform of 64-bit CRC circuit is shown in figure 6.



This is the Frame Check Sequence that is to be added along with the message

Fig 6: Simulation result of 64-bit CRC circuit

The Synthesis Report presented in Table shown below. From the device utilization summary Only 29% utilization of Flip flop is done. So on the same FPGA chip 128 bit ,256 bit CRC can also be implemented.

Logic Utilization	Used Resource	Available Resource	% utilizat
Number of slice flip flops	563	1,920	29%
Number of 4 input LUTs	195	1,920	10%
Number of occupied Slices	285	960	29%
Number of bonded IOBs	51	66	77%
Number of BUFGMUXs	2	24	8%
Number of Slices containing only related logic	285	285	100%
Number of Slices containing only related logic	0	285	0%

Fig 7: Synthesis report for 64-bit CRC Circuit

### On-Chip Debugging using CHIPSCOPE

Driving signals to external I/O introduces additional problems like spurs glitches etc. So this is an Inflexible solution and also it is difficult or impossible to add additional debug pins if needed. Ultimately we will have only limited visibility to on-chip activities. ChipScope is an embedded, software based logic analyzer. By inserting an “Integrated CONTroller core” (ICON) and an “Integrated Logic Analyzer” (ILA) into the design and connecting them properly, we can monitor any or all of the signals in the design. ChipScope provides us with a convenient software based interface for controlling the “integrated logic analyzer,” including setting the

triggering options and viewing the waveforms. ChipScope Pro has an Integrated Logic Analyzer Core. So, No I/O Pins Required For Debugging. The complete access of the FPGA resources is done only using the JTAG Port. This provides an easy way to On-Chip access to every signal and node in the FPGA Design with using JTAG pins and eliminates the need for extensive dedicated I/O. Driving Signals reduces all additional problems. It is easy to add and remove Cores at any time in the Design Process providing complete analysis and debugging solution. These ChipScope modules are extremely useful because they allow us to view and manipulate signals directly from hardware during run-time Touch the particular trigger (Reset button) on the hardware board to run the design, to respective buses and then hit the play button. It can be observed that the entire output signals are varied and we can do the verification. The Bus Plot waveform for the CRC circuit is shown in Fig.8.

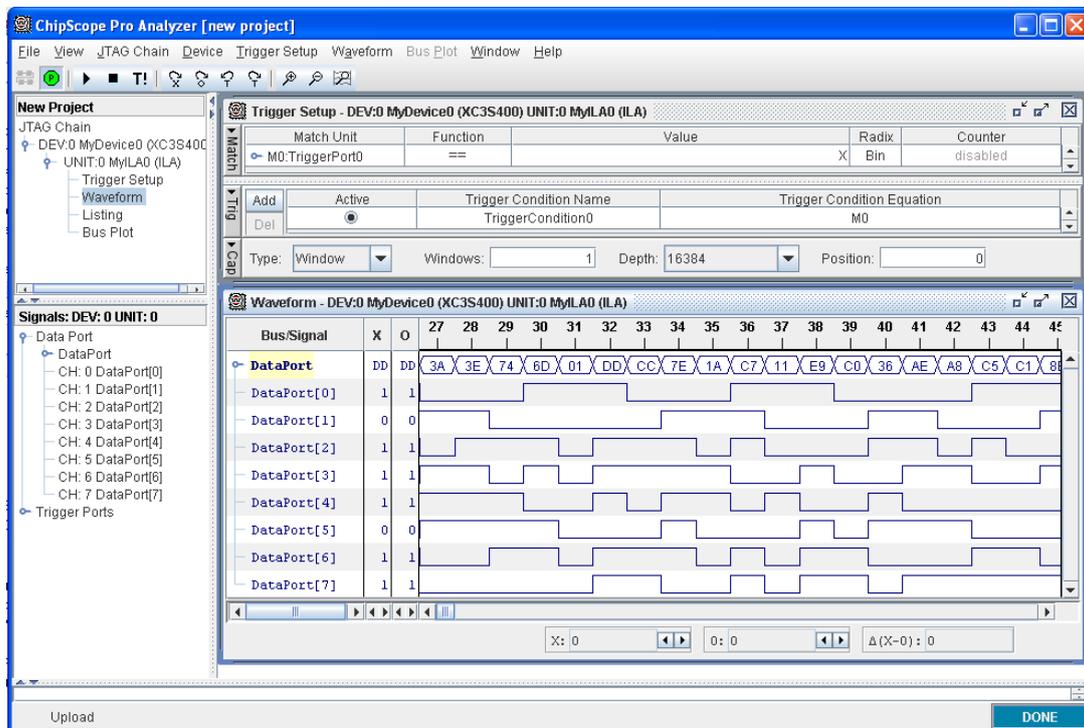


Fig 8 : Bus Plot for CRC

### VIII. Conclusions

The 64-bit field programmable CRC computation circuit is designed, synthesized, and implemented using Xilinx ISE Foundation 10.1i which can be used in Data communication Networks. For Behavioral simulation, Place and Route simulation Modesim6.0 is used. This system has the features of CRC parameters were fully programmable. This was achieved by deriving an array of processing cells to implement a matrix based computation technique.64-bit CRC circuit provides high flexibility while allowing high performance at a lower hardware cost.

### Future Scope

The architecture is also generic in its design and can be scaled to 64-, 128-, or 256-bits in the data path, enabling support of throughput rates up to 40 Gb/s at 256-bits. It is further anticipated that a physical oriented design methodology, such as a data-path compiler can be used to optimize the regular structure of the programmable cell array, which could significantly increase the operational frequency while maintaining a low hardware cost. Such an optimized circuit represents an attractive hard macro for environments requiring low cost hardware flexibility, and in emerging areas such as iSCSI-based SANs, where the flexibility to adopt emerging protocols offers a key advantage to vendors.

References

- [1] Data Communications and Networking , Third Edition By Behrouz A.Forouzan.
- [2] P. Koopman and T. Charkravarty, “Cyclic Redundancy Code (CRC) polynomial selection for embedded networks,” in Proc. DSN, pp. 145–154.
- [3] Design and Implementation of a Field Programmable CRC Circuit Architecture, Ciaran Toal, Kieran cLaughlin, Sakir Sezer, and Xin Yang
- [3] P. Koopman, “32-bit cyclic redundancy codes for internet applications,” in Proc. DSN, pp. 459–472.640–651, Dec. 1995.
- [5] T. Bi-Pei and C. Zukowski, “High-speed parallel CRC circuits in VLSI,” IEEE Trans. Commun., vol. 40, no. 4, pp. 653–657, Apr. 1992.
- [6] J. H. Derby, “High-speed CRC computation using state-space transformations,” in Proc. Globecom, Nov. 2001, pp. 166–170.
- [7] M.-D. Shieh, M.-H. Sheu, C.-H. Chen, and H.-F. Lo, “A systematic approach for parallel CRC computations,” J. Inf. Sci. Eng., vol. 17, pp.445–461, 2001.
- [8] T. Henriksson and D. Liu, “Implementation of fast CRC calculation,” in Proc. ASP-DAC, 2003, pp. 563–564.
- [9] 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. ICECS, 2001, vol. 3, pp. 1231–1234.
- [10] O. Weiss, M. Gansen, and T. Noll, “A flexible data path generator for physical oriented design,” in Proc. ESSCIRC, Villach, Sep. 2001, pp. 408–411.
- [11] Data Communications and Teleprocessing Systems by Trevor Housley, Second Edition

<p><b>G.Shanthi</b> did her B.Tech in Electronics and Communication Engineering from JNTUH College, Hyderabad and obtained her Masters degree in VLSI system Design from VNR Vignan Jyothi Institute of engineering and Technology, Hyderabad.</p> <p>She has 10 years of working experience and presently working as Lecturer in Electronics and communication Engg. Department in VNR Vignan Jyothi Institute of engineering and Technology, Hyderabad.. Her areas of research include Data communications, Error Detecting Methods</p>	
<p><b>L.Padma Sree</b> did her B.Tech in Electronics and Communication Engineering from Bapatla Engineering college, Nagarjuna University and obtained her Masters degree in Digital systems &amp; computer electronics from J.N.T.U., Ph.D in Electronics and Communication Engineering from JNTUH .</p> <p>She has 15 years of Teaching experience and presently working as Professor in Electronics and communication Engg. Department in VNR Vignan Jyothi Institute of engineering and Technology, Hyderabad.. Her areas of research include Digital Image Processing, Computer Security, Communications.</p>	