A High Performance CRC Checker for Ethernet Application

Deepti Rani Mahankuda & M. Suresh

Electronics and Communication Engineering Dept.
National Institute of Technology, Berhampur, Odisha, India.
E-mail:deepti.rani07@gmail.com

Abstract **: This Project presents the generation of Cycle Redundancy Check (CRC) in the field of Data Communication and Networking. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. It is for the purpose of error detection in communication networks. This project primarily focuses on error detection in the Ethernet applications. The main object of this project is to implement high performance error detection technique in a 32 bit Parallel data sending.**

Keywords-CRC, Computation of CRC Polynomial, Verilog and CADENCE Tool.

I. Introduction

CRCs are based on the theory of cyclic error-correcting codes. The use of systematic cyclic codes, which encode messages by adding a fixed-length check value, for the purpose of error detection in communication networks was first proposed by W. Wesley Peterson in 1961. Cyclic codes are not only simple to implement but have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages. This is important because burst errors are common transmission errors in many communication channels, including magnetic and optical storage devices. 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.

Specification of a CRC code requires definition of a so-called generator polynomial. This polynomial resembles the divisor in a polynomial long division, which takes the message as the dividend and 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 generator polynomial, which therefore determines how long the result can be.

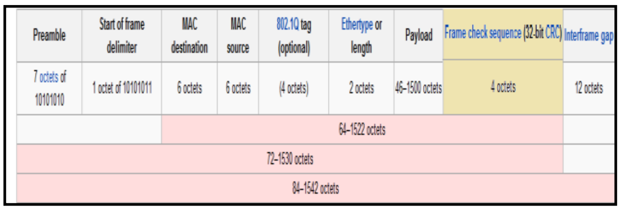
In practice, all commonly used CRCs employ the finite field GF (2). This is the field of two elements, usually called 0 and 1, comfortably matching computer architecture. The simplest error-detection system, the parity bit, is in fact a trivial 1-bit CRC: it uses the generator polynomial x+1.

II. Description

The design of the 32-bit polynomial most commonly used by standards bodies, CRC-32-IEEE, was the result of a joint effort for the Rome Laboratory and the Air Force Electronic Systems Division by Joseph Hammond, James Brown and Shyan-Shiang Liu of the Georgia Institute of Technology and Kenneth Brayer of the MITRE Corporation. The earliest known appearances of the 32-bit polynomial were in their 1975 publications: Technical Report 2956 by Brayer for MITRE, published in January and released for public dissemination through DTIC in August, and Hammond, Brown and Liu's report for the Rome Laboratory, published in May. Both reports contained contributions from the other team. In December 1975, Brayer and Hammond presented their work in a paper at the IEEE National Telecommunications Conference: the IEEE

CRC-32 polynomial is the generating polynomial of a Hamming code and was selected for its error detection performance. Even so, the Castagnoli CRC-32C polynomial used in iSCSI or SCTP matches its performance on messages from 58 bits–131 kbits, and outperforms it in several size ranges including the two most common sizes of Internet packet. The ITU-T G.hn standard also uses CRC-32C to detect errors in the payload (although it uses CRC-16-CCITT for PHY headers).

III. Ethernet Frame



Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype or length	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets of 10101010	1 octet of 10101011	6 octets	6 octets	(4 octets)	2 octets	46–1500 octets	4 octets	12 octets
		64–1522 octets						
72–1530 octets								
84–1542 octets								

The commonly used polynomial for Ethernet is given by:

$$\mathbf{x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1}$$

We will use this polynomial for all further calculations involved.

A. *Serial CRC Checker:*

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. By this form we will conclude where the data is accurate or corrupted.

*A .1.*Verification and Simulation

In this part the test bench and simulation result will be shown. In this test bench the signals will be generated and the data will be given to the input and the output will be shown. And finally the simulation result will be displayed by Simvision tool. The simulation result is given below. As in the above figure the simulation result is given as zero. It means CRC value is zero. So the exact data has entered. Simulation has completed at 4040 ns.(Fig. 1)

A.2. Synthesis

Synthesis process will be done by using Cadence RTL Compiler (RC) tool. The process will generate area report, power report, timing report, clock gating report. The generated output will be the net list file and sdc file (Fig. 2)

Refer to Table 1(b),2(b),3(b) of Serial CRC Checker.

B. *Parallel CRC Checker*

In this implementation, the data input will be 16 bits (2 Byte) at one clock pulse so the data input size will be increase so clock pulse will be decreased. It will complete the result in lesser clock pulse so it will be used for high performance. There will be various processes to implement the parallel crc checking process.

B.1. Design Procedure

In this procedure there will be two matrixes will be generated as H1 and H2 matrix. There is Nin will be data input and Min CRC input, N and M will be data width and CRC width. During generation of H1 matrix Nin will be one hot encoded and Min will be zero. In H2 matrix Nin will be zero and Min will be one hot encoded. For the parallel CRC generation, there will be pseudo code as CRC parallel. In this code there will be a regular iteration with the data size and CRC polynomial size. And finally the variables where the matrix is 1 that will be Ex-ORed.

B.2. Verification and Simulation

As the serial CRC implementation, the parallel CRC implementation the signals will be generated and the output is verified. The data input has difference as the data input is not one bit, it is 16 bit input in one clock pulse. The data input is 16 bits or 2 byte and the final output is zero. The simulation result is given below. (Fig 3)

B.3.Synthesis

Synthesis process will be done by using Cadence RTL Compiler (RC) tool. The process will generate area report, power report, timing report, clock gating report. The generated output will be the net list file and sdc file (Fig. 4)

Refer to Table 1(a),2(a),3(a) of Parallel CRC Checker.

IV. Final Layout Design

This is the final and optimized layout of the Parallel CRC Checker. In this layout there are the standard cells have used and the cells are connected through the metals and it has optimized before routed and after routed. (Fig. 5)

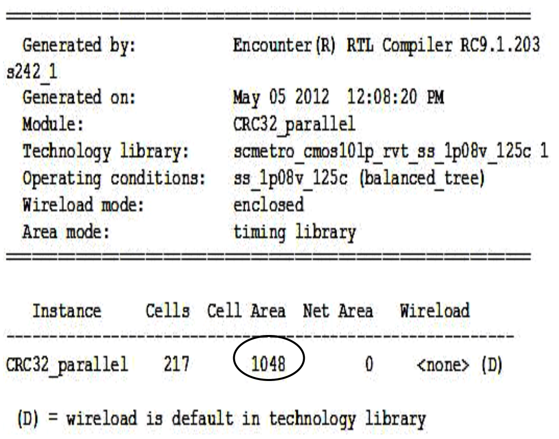
V. Comparision Of Serial And Parallel Crc

Both Serial and Parallel CRC Checker for Ethernet has been designed and synthesized. Now the comparison will be done between this two designs. The comparison will be done of the simulation results, area, timing and power.(Fig 6,7)

VI. Simulation

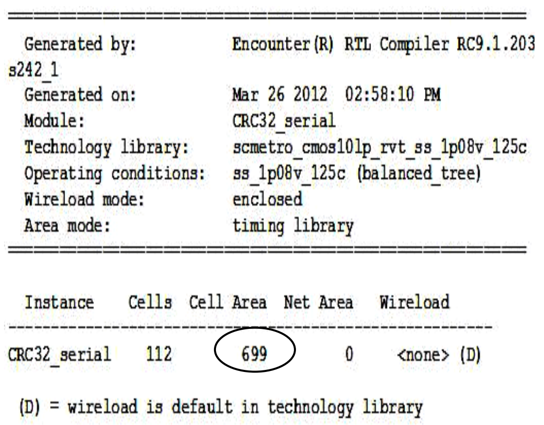
In the above two simulation result it has been observed that serial simulation has been completed at 4040 ns whereas parallel simulation has completed at 300 ns. So we have completed the same task very less clock pulse. Parallel CRC Checker completes its work 3740 ns before serial CRC checker completes.(Fig. 6,7as compared in Fig. 1,3)

Area Reports



```
====================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203
s242_1
  Generated on:           May 05 2012  12:08:20 PM
  Module:                 CRC32_parallel
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c 1
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================

   Instance     Cells  Cell Area  Net Area   Wireload
-------------------------------------------------------
CRC32_parallel   217     1048         0     <none> (D)

  (D) = wireload is default in technology library
```

Table 1. (a) Parallel CRC Checker area report



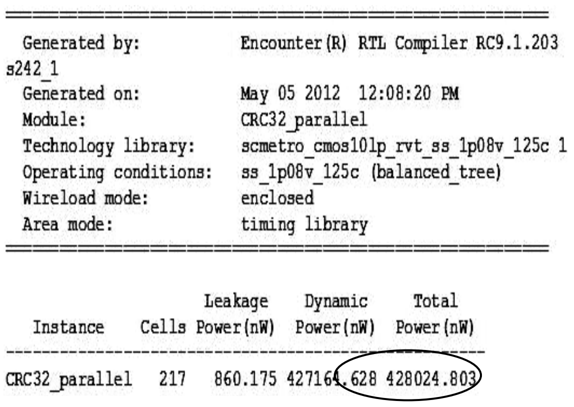
```
====================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203
s242_1
  Generated on:           Mar 26 2012  02:58:10 PM
  Module:                 CRC32_serial
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================

  Instance    Cells  Cell Area  Net Area    Wireload
-----------------------------------------------------
CRC32_serial    112      699          0     <none> (D)

  (D) = wireload is default in technology library
```

Table 1(b) Serial CRC Checker

Area report

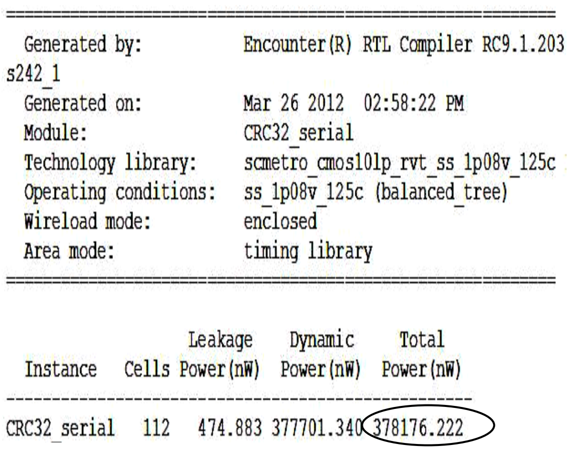
In this both area reports, we have observed that Parallel CRC Checker has cell area 1048 nm2 Serial CRC Checker 699 nm2. And the cells have been increased 112(Serial CRC Checker) to 217(Parallel CRC Checker).

Power Report:



```
=========================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203
s242_1
  Generated on:           May 05 2012  12:08:20 PM
  Module:                 CRC32_parallel
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c 1
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=========================================

                        Leakage   Dynamic     Total
   Instance     Cells Power(nW)  Power(nW)  Power(nW)
------------------------------------------------------
CRC32_parallel   217   860.175 427164.628 428024.803
```

Table 2 (a) Parallel CRC Checker;

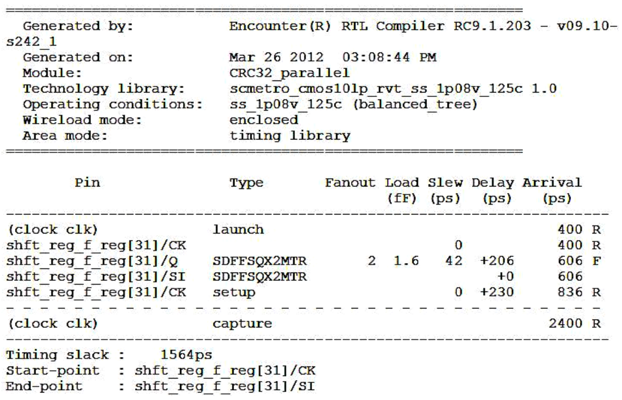


```
=========================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203
s242_1
  Generated on:           Mar 26 2012  02:58:22 PM
  Module:                 CRC32_serial
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=========================================

                     Leakage   Dynamic     Total
  Instance    Cells Power(nW)  Power(nW)  Power(nW)
----------------------------------------------------
CRC32_serial   112   474.883 377701.340 378176.222
```

Table2 (b) Serial CRC Checker

As the cells have reduced of Serial CRC Checker compared to Parallel CRC Checker. So the total power has reduced from 428024.803 nW (Parallel CRC Checker) to 378176.222 nW (Serial CRC Checker).

Timing Report



```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203 - v09.10-
s242_1
  Generated on:           Mar 26 2012  03:08:44 PM
  Module:                 CRC32_parallel
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c 1.0
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

        Pin              Type       Fanout Load Slew Delay Arrival
                                           (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------------
(clock clk)             launch                                400 R
shft_reg_f_reg[31]/CK                              0          400 R
shft_reg_f_reg[31]/Q    SDFFSQX2MTR      2  1.6   42  +206    606 F
shft_reg_f_reg[31]/SI   SDFFSQX2MTR                     +0    606
shft_reg_f_reg[31]/CK   setup                      0  +230    836 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)             capture                              2400 R
-------------------------------------------------------------------
Timing slack :    1564ps
Start-point  : shft_reg_f_reg[31]/CK
End-point    : shft_reg_f_reg[31]/SI
```

```
=============================================================
  Generated by:           Encounter(R) RTL Compiler RC9.1.203 - v09.10-
s242_1
  Generated on:           Mar 26 2012  02:58:16 PM
  Module:                 CRC32_serial
  Technology library:     scmetro_cmos10lp_rvt_ss_1p08v_125c 1.0
  Operating conditions:   ss_1p08v_125c (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

        Pin               Type       Fanout Load Slew Delay Arrival
                                            (fF) (ps)  (ps)    (ps)
---------------------------------------------------------------------
(clock clk)             launch                                 400 R
shft_reg_f_reg[30]/CK                               0          400 R
shft_reg_f_reg[30]/Q    SDFFSQX2MTR        2  1.6   42  +206   606 F
shft_reg_f_reg[30]/SI   SDFFSQX2MTR                       +0   606
shft_reg_f_reg[30]/CK   setup                       0   +230   836 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)             capture                               2400 R
---------------------------------------------------------------------
Timing slack :    1564ps
Start-point  : shft_reg_f_reg[30]/CK
```

Table3 (a) Parallel CRC Checker; (b) Serial CRC Checker

From the above comparison, it has been observed that the clock pulse in the simulation has been reduced in Parallel CRC Checker. But the no of cells, area size and power have been increased in Parallel CRC Checker as compared to Serial CRC Checker. And the timing slack is same. Although Parallel CRC Checker has increased in area, power but it reduces the simulation time. So it is called as high performance Checker.

VII. Conclusion

From all the analysis of both Serial and Parallel implementation of CRC Checker, it has been concluded that Parallel CRC Checker for Ethernet is high speed and high performance Checker. Although Parallel CRC Checker has larger in area and power as compared to Serial CRC Checker but it reduces the simulation time or checking time at the optimum level which is required in High performance application. So it will very much suitable for high performance checking process for Ethernet application.

Acknowledgment

We are thankful to National Institute of Science and Technology Management for providing us the oppurtunity to work on this project

References

[1] "Data Communication and Networking" 4th Edition by "Behrouz A Forouzan" published by "The McGraw Hills Company"

[2] W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection, *"Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[3] Jun-Ren Chen, Incorporating data abstraction in CRC to simplify method interface, International Computer Symposium (ICS), 2010 , pp- 983 – 987, Dec, 2010

[4] http://www.mhhe.com/forouzan/dcn4sie

[5] http://www.circuitcellar.com/network.

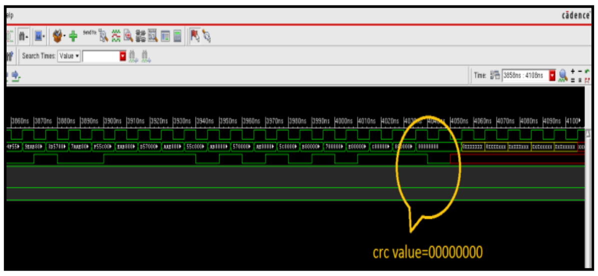


Figure 1. Simulation result of Serial CRC Checker

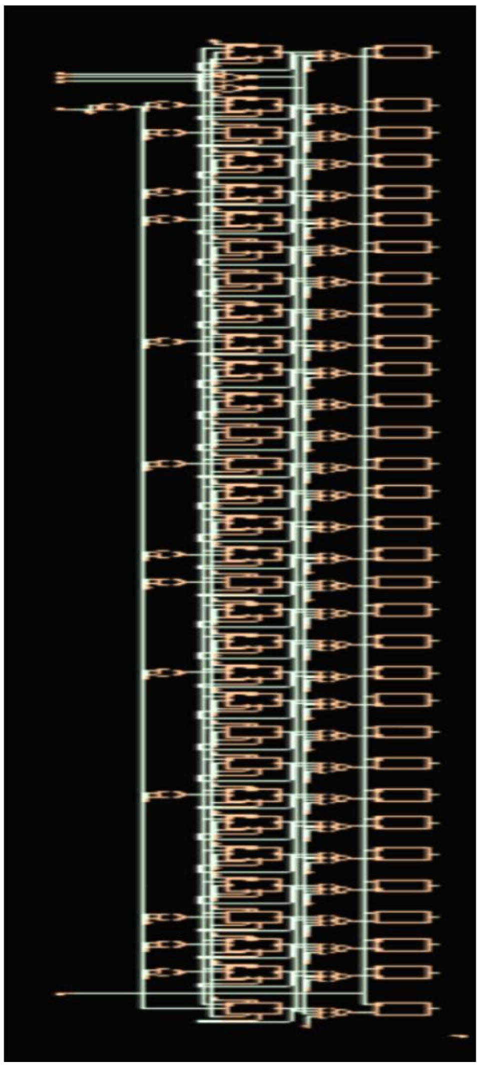


Figure 2. : Synthesize circuit of Serial CRC RTL

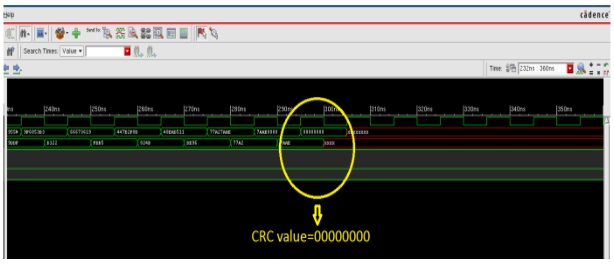


Figure 3. : Simulation result of Parallel CRC Checker

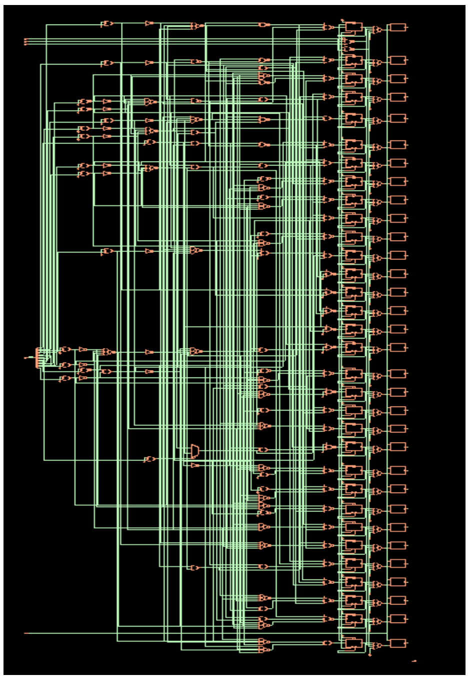


Figure 4. Synthesize Output of Parallel CRC RTL

Figure 5.

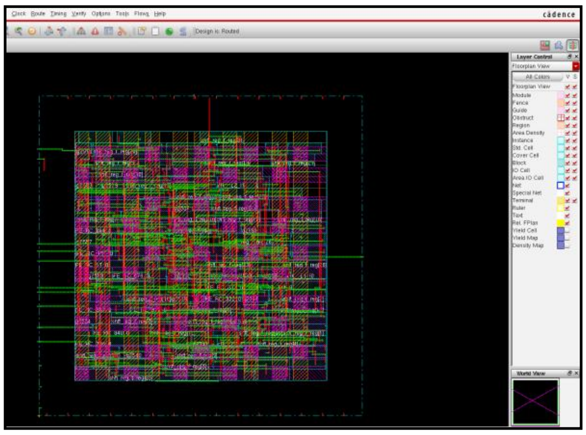


Figure 6. Layout of the Circuit

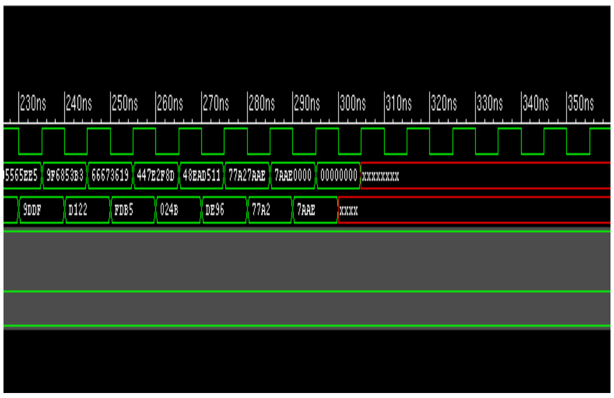


Figure 7. Parallel CRC Checker

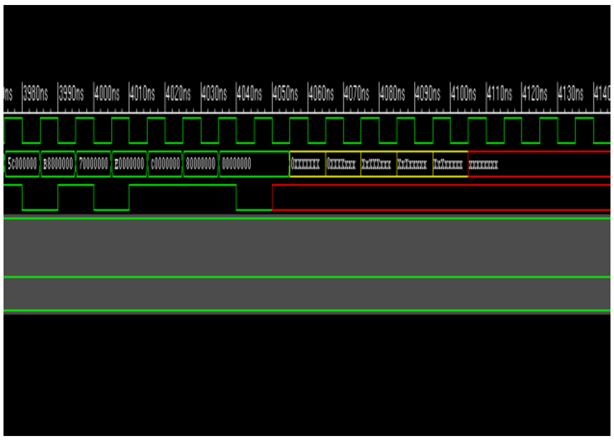


Figure 8. Serial CRC Checker

❑❑❑