CRC Coding

By: Sudhamshu B N(1DS18EC091)

Rajasudhan Gowda S A(1DS19EC426)

Nazakat Nazir Sofi(1DS18EC58)

Submitted to: Prof.Madhura R

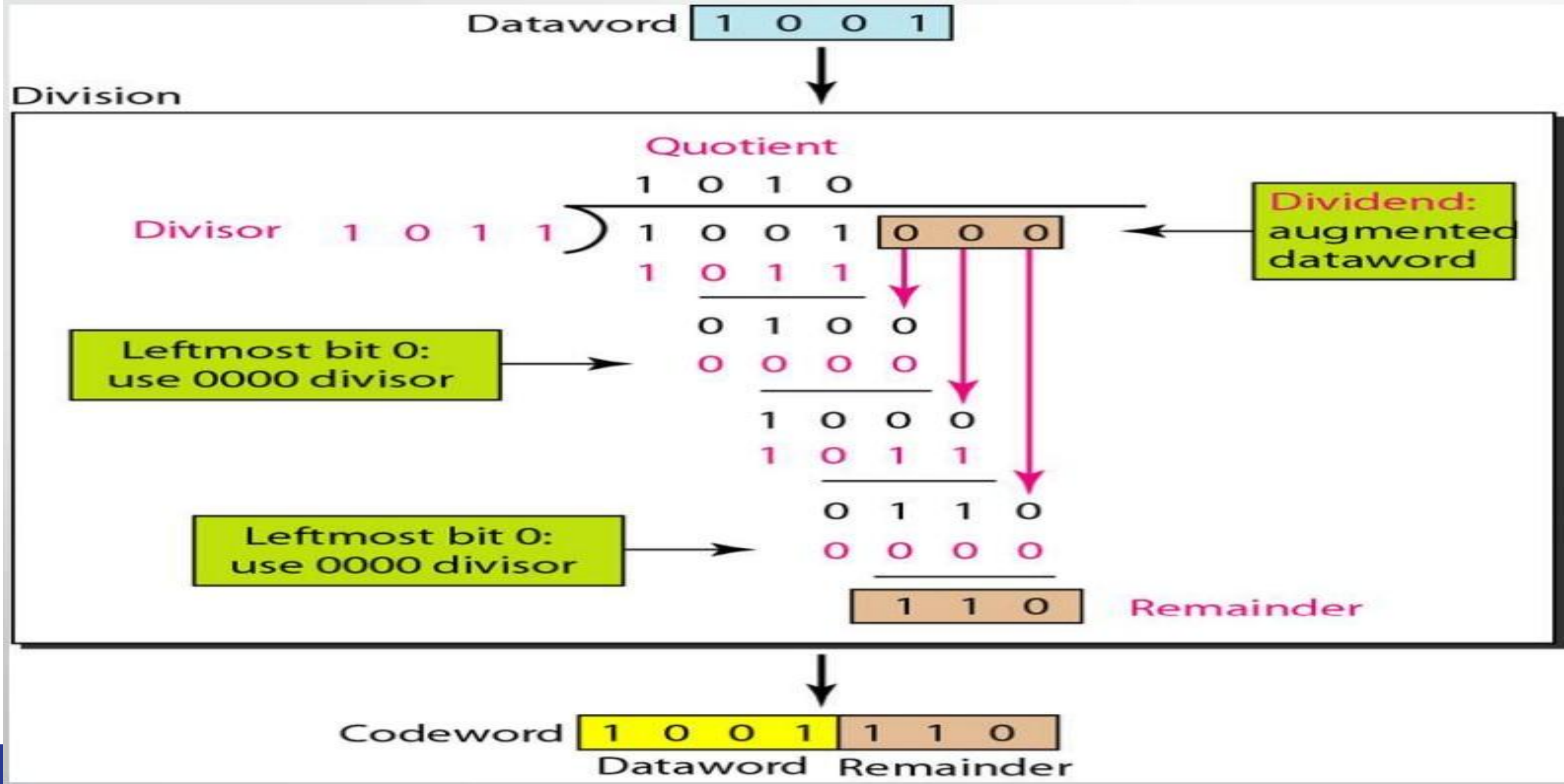
Introduction

- CRC is a widely used technique for error checking in streams of data
- Used in protocols used in data transmission.
- CRC-16 and CRC-32 with user defined proprietary polynomials
- ModelSim Software for the simulation.
- Simulating CRC 16 Serial, CRC 16 Parallel, CRC 32 Serial and CRC 32 Parallel for the given polynomial.

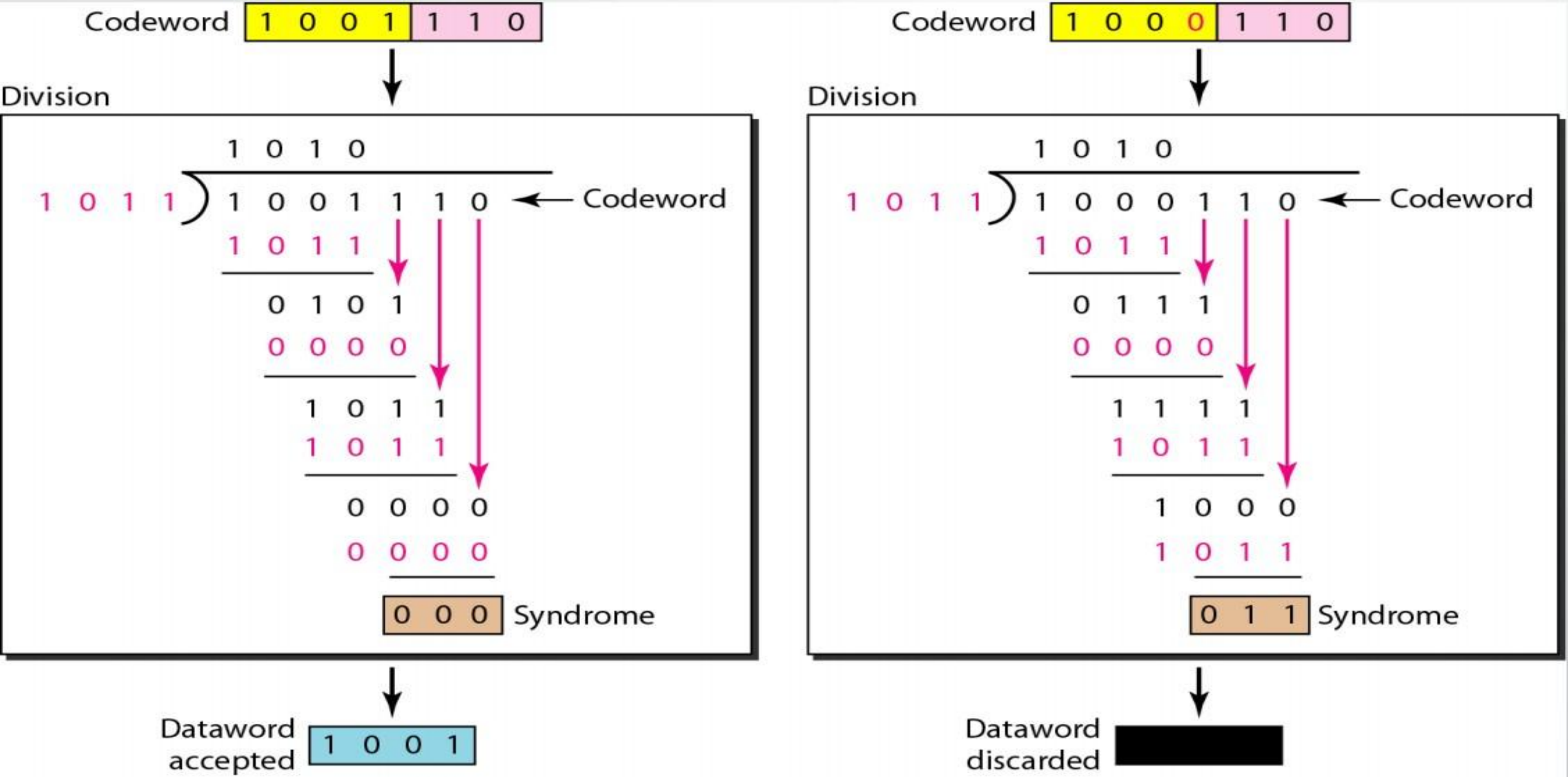
What is CRC?

- Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a code word is cyclically shifted (rotated), the result is another code word.
- A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.
- Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

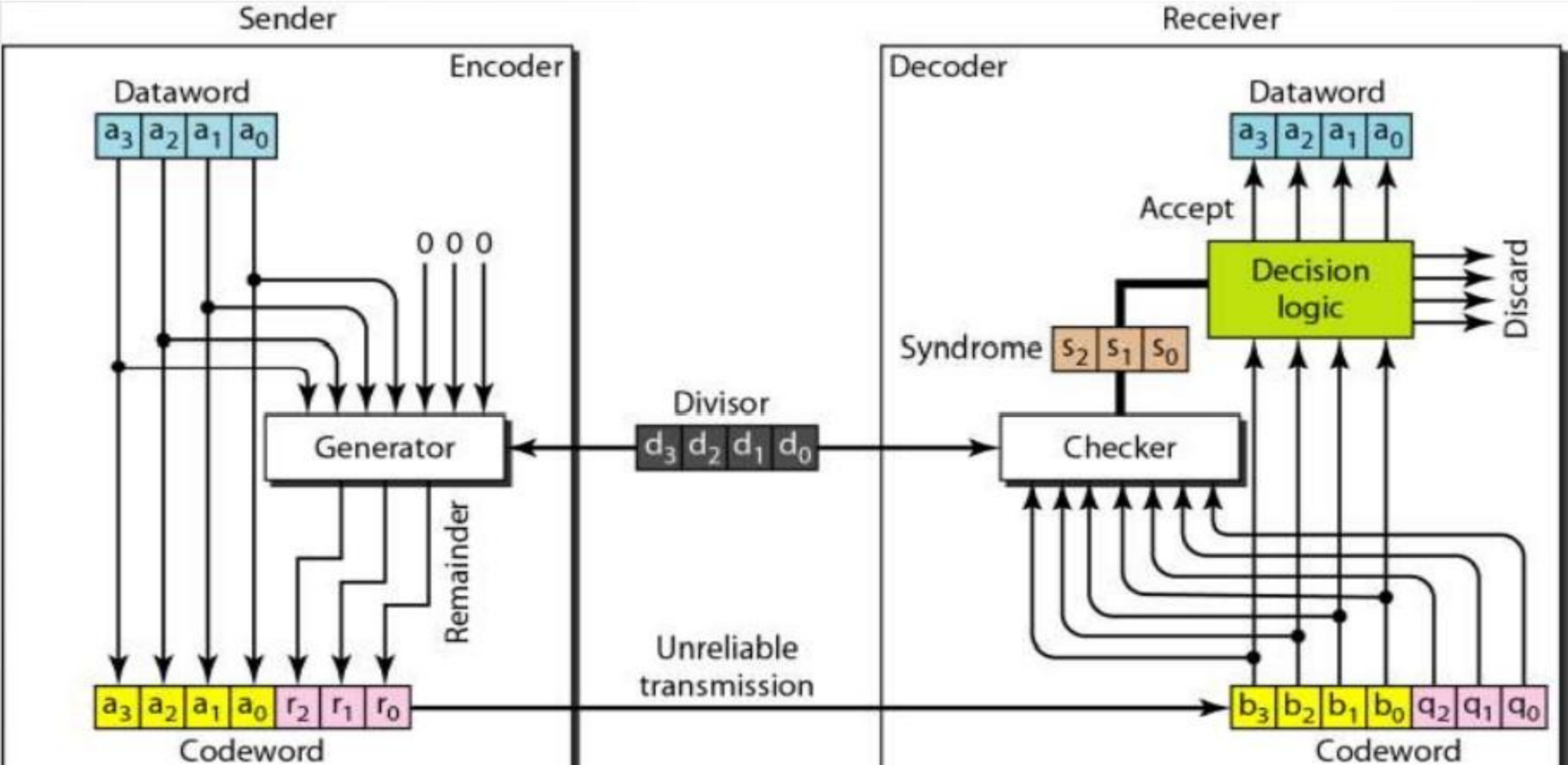
Division in CRC Encoder



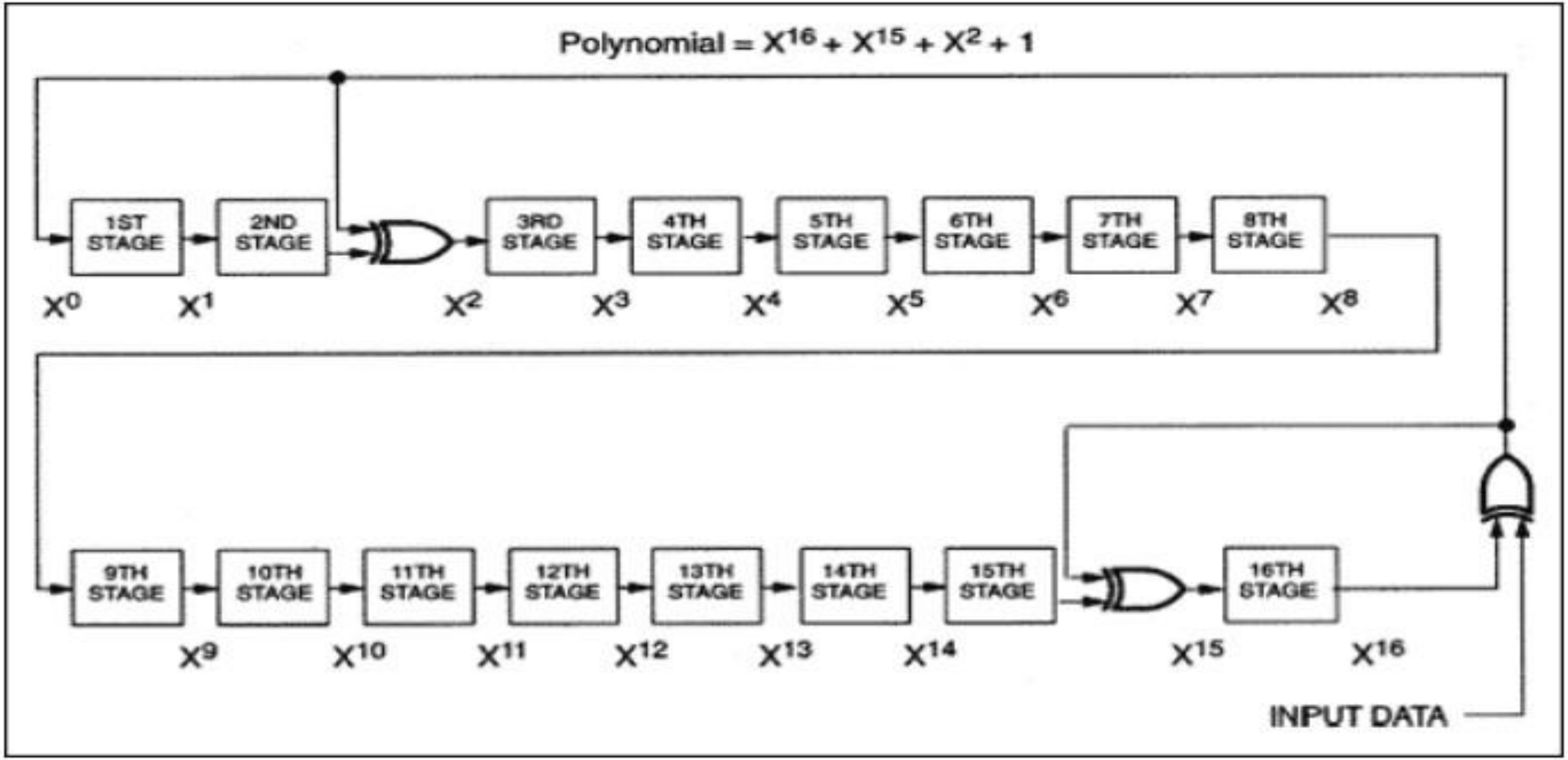
Division in Decoder



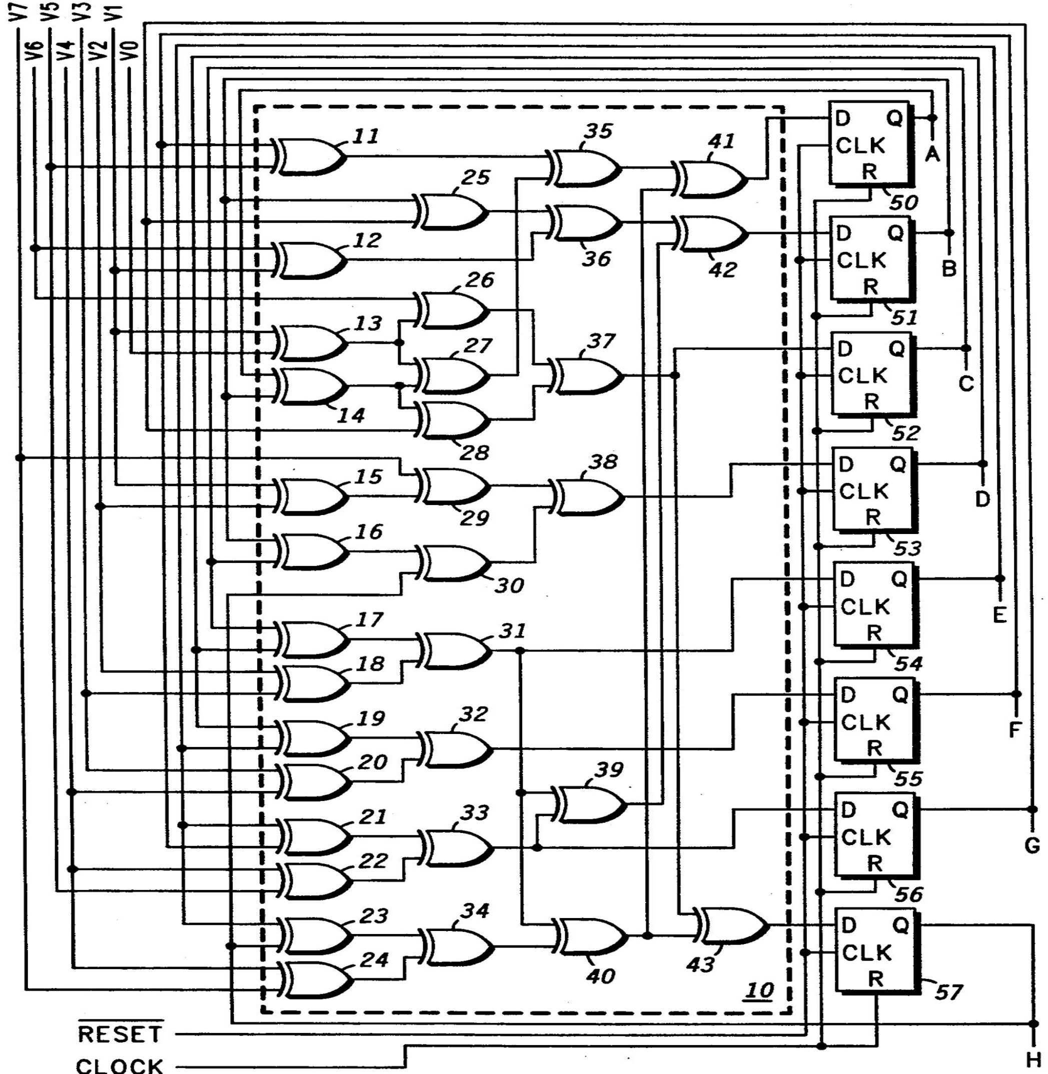
CRC Encoder and Decoder



CRC 16 bit serial example



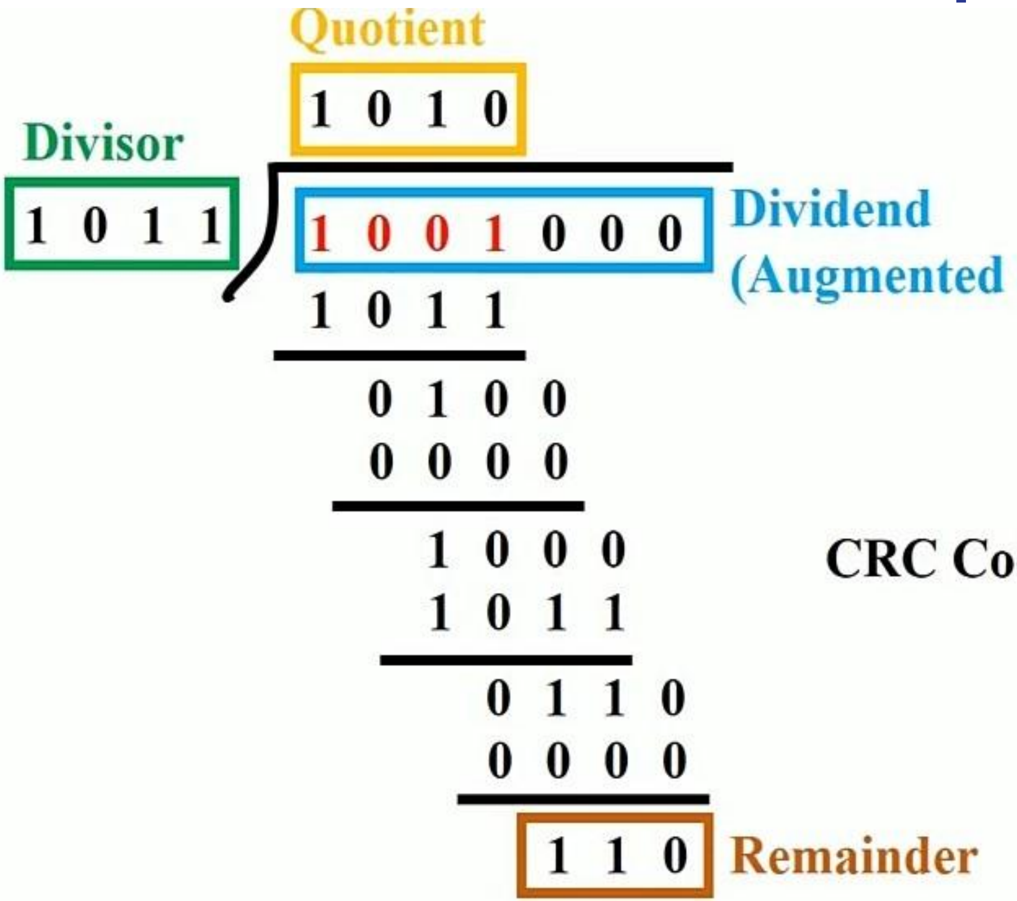
CRC 8 bit Parallel Example



Hardware Implementation Series

- k = the length of dataword
- r = the length of redundant bits
- n = k+r = the length of codeword
- 🟧 = Shift Register
- ⊕ = XOR Gate

Concept


Quotient

1 0 1 0

Divisor

1 0 1 1

1 0 0 1 0 0 0

Dividend (Augmented Codeword)

```
1 0 1 1
-------
  0 1 0 0
  0 0 0 0
  -------
    1 0 0 0
    1 0 1 1
    -------
      0 1 1 0
      0 0 0 0
      -------
        1 1 0
```

Remainder

k = 4

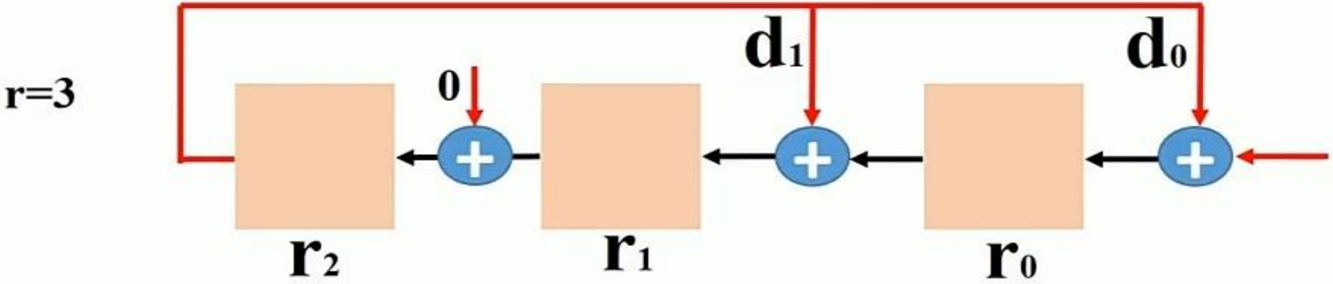
r = 3

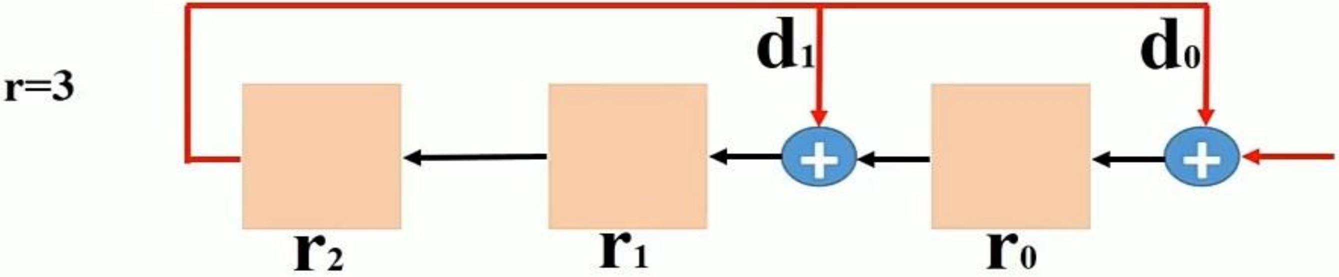
n = k+r = 4+3=7

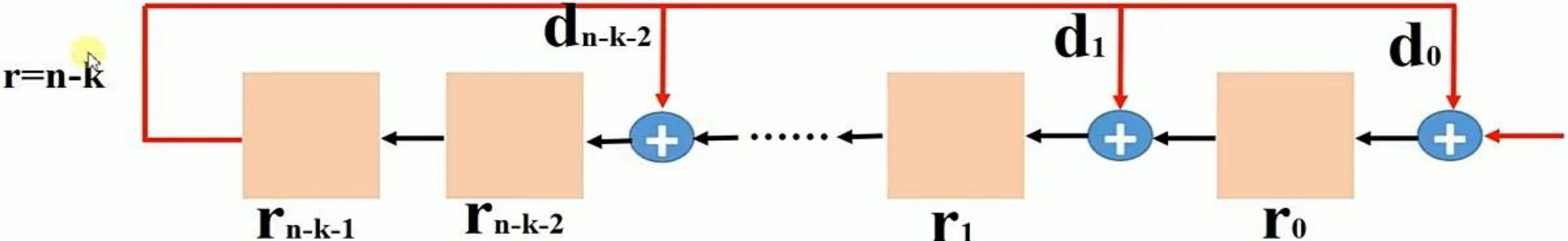
Dataword Remainder

CRC Codeword→ 1 0 0 1 1 1 0

Principle And Implementation







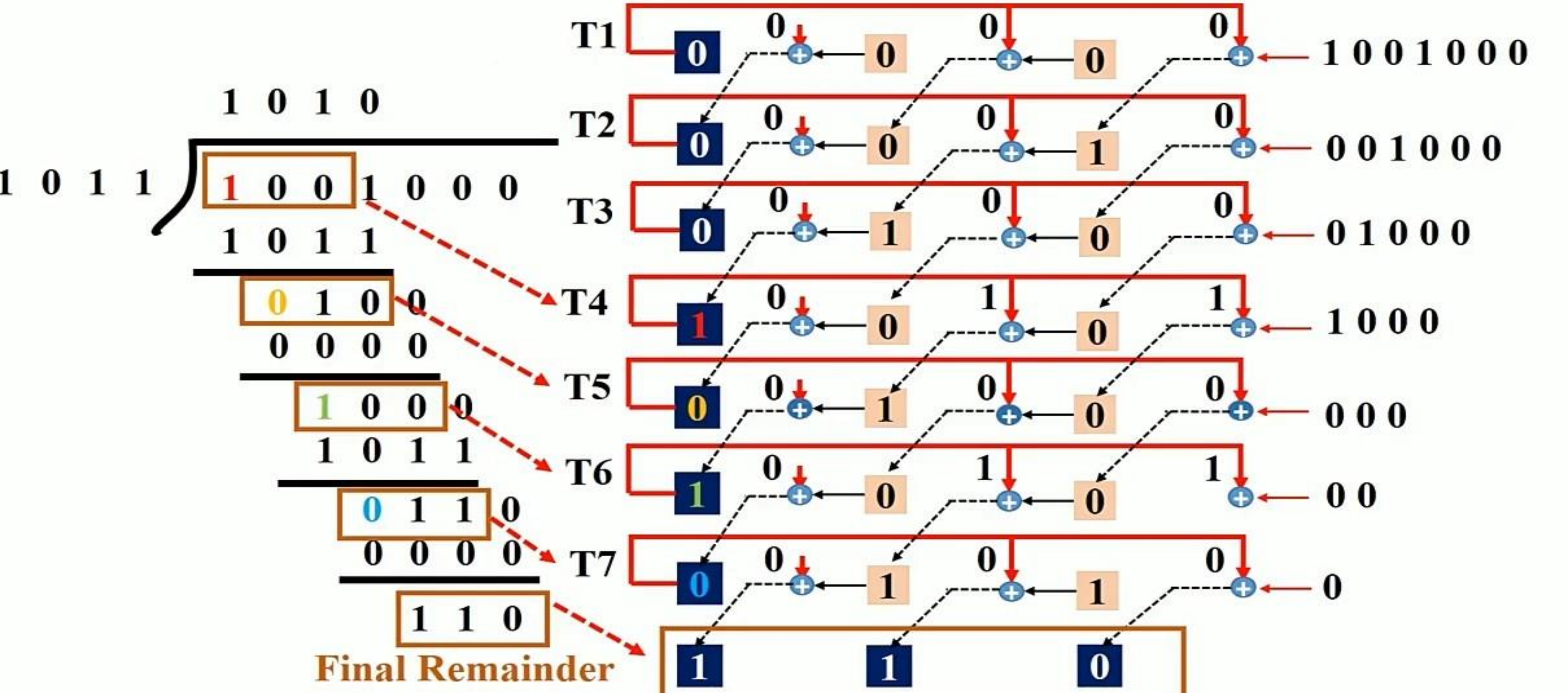
4 bit Series implementation

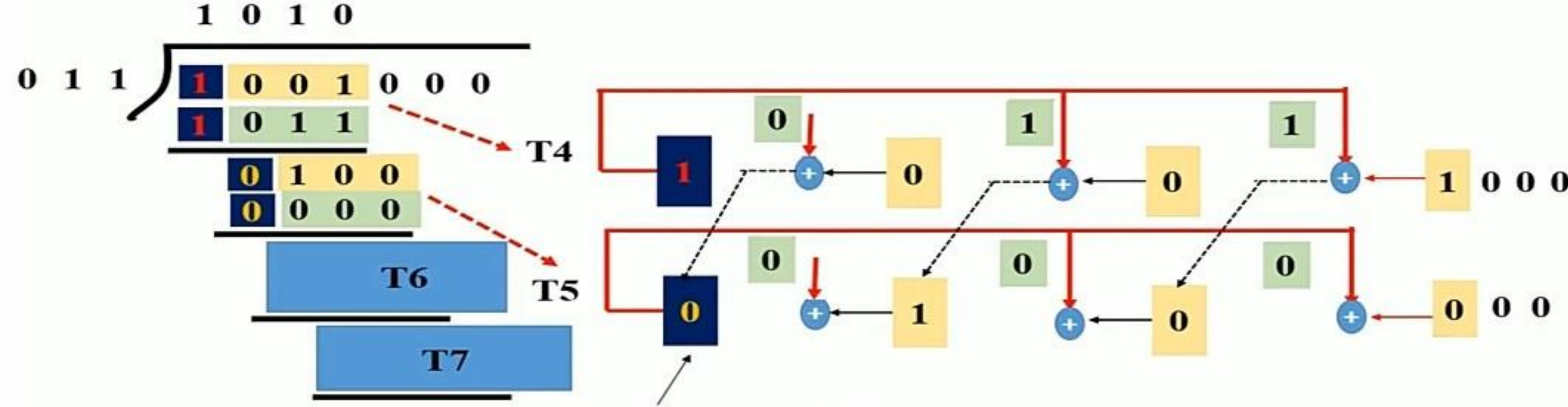
T1 → one clock time

⊕ → XOR

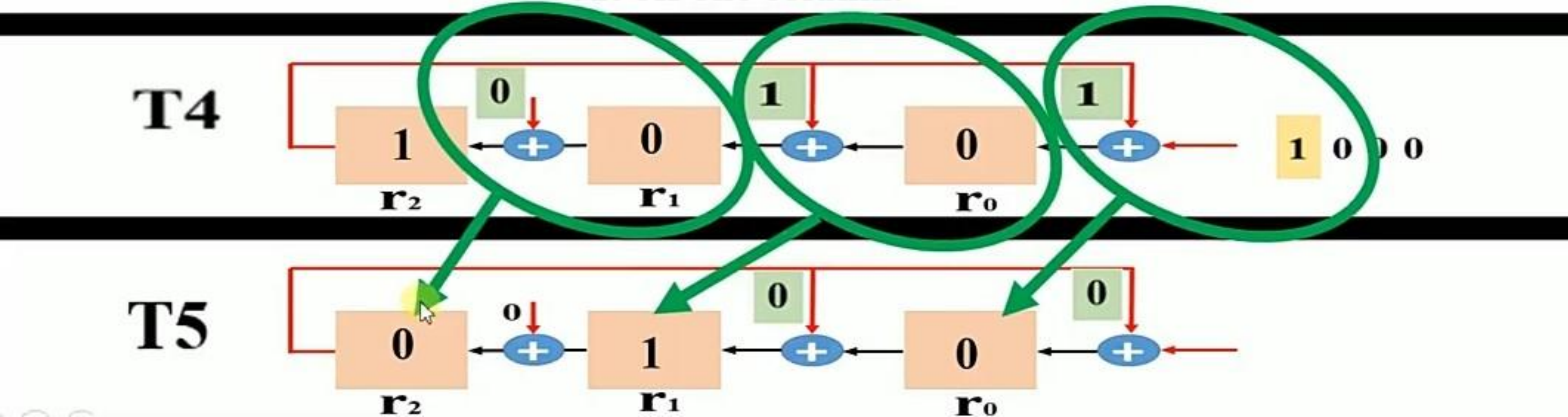
▨ → Shift Register (Remainder)

Augmented dataword





Each time we will cut off one column.

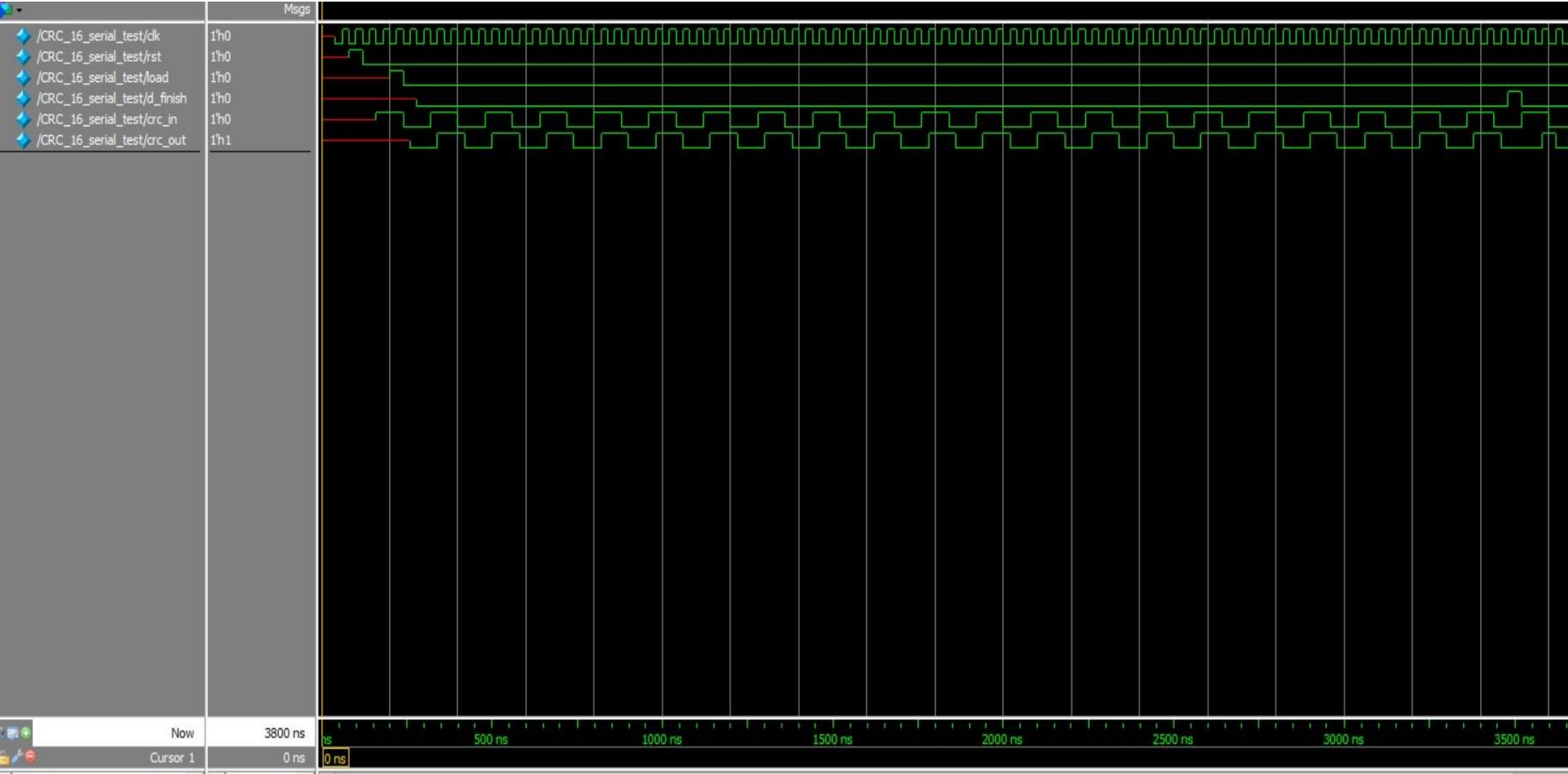


Mini project Files

sudhamshu091/Verilog (github.com)

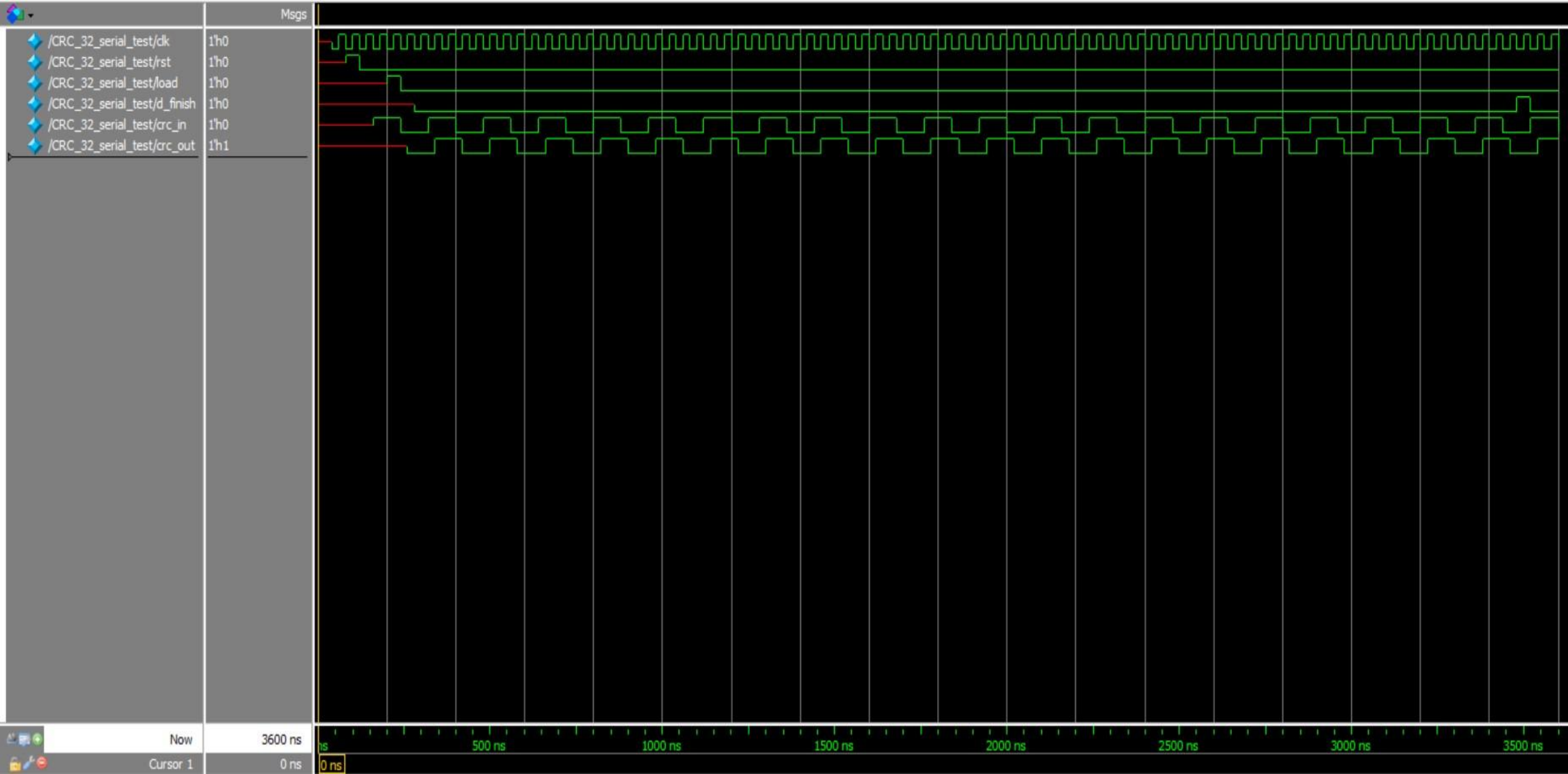
Verilog/CRC Coding · sudhamshu091/Verilog (github.com)

Results



	Msgs
/CRC_16_serial_test/clk	1'h0
/CRC_16_serial_test/rst	1'h0
/CRC_16_serial_test/load	1'h0
/CRC_16_serial_test/d_finish	1'h0
/CRC_16_serial_test/crc_in	1'h0
/CRC_16_serial_test/crc_out	1'h1
Now	3800 ns
Cursor 1	0 ns

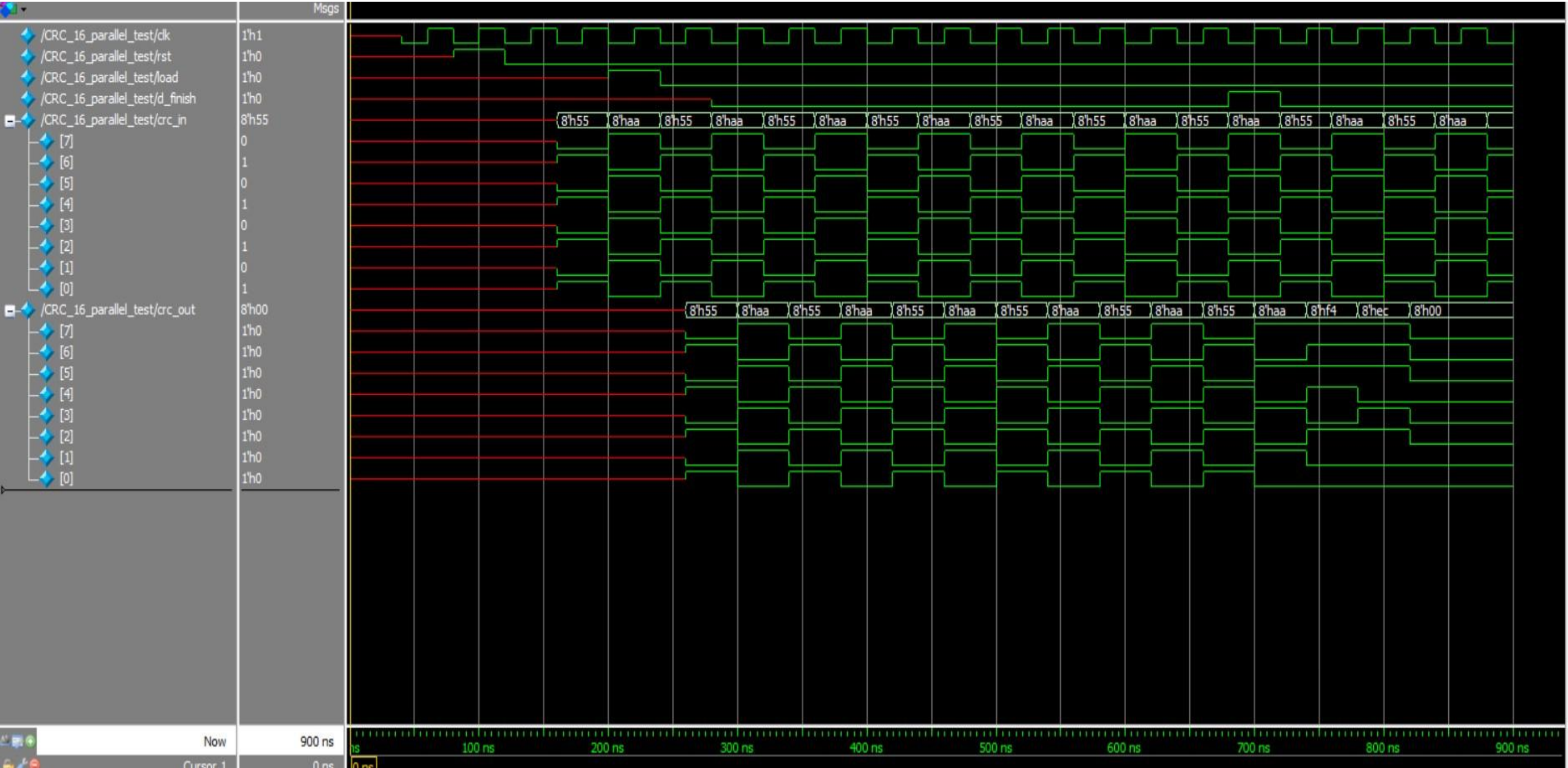
Results



```
/CRC_32_serial_test/clk       1'h0
/CRC_32_serial_test/rst       1'h0
/CRC_32_serial_test/load      1'h0
/CRC_32_serial_test/d_finish  1'h0
/CRC_32_serial_test/crc_in    1'h0
/CRC_32_serial_test/crc_out   1'h1

Now        3600 ns
Cursor 1   0 ns
```

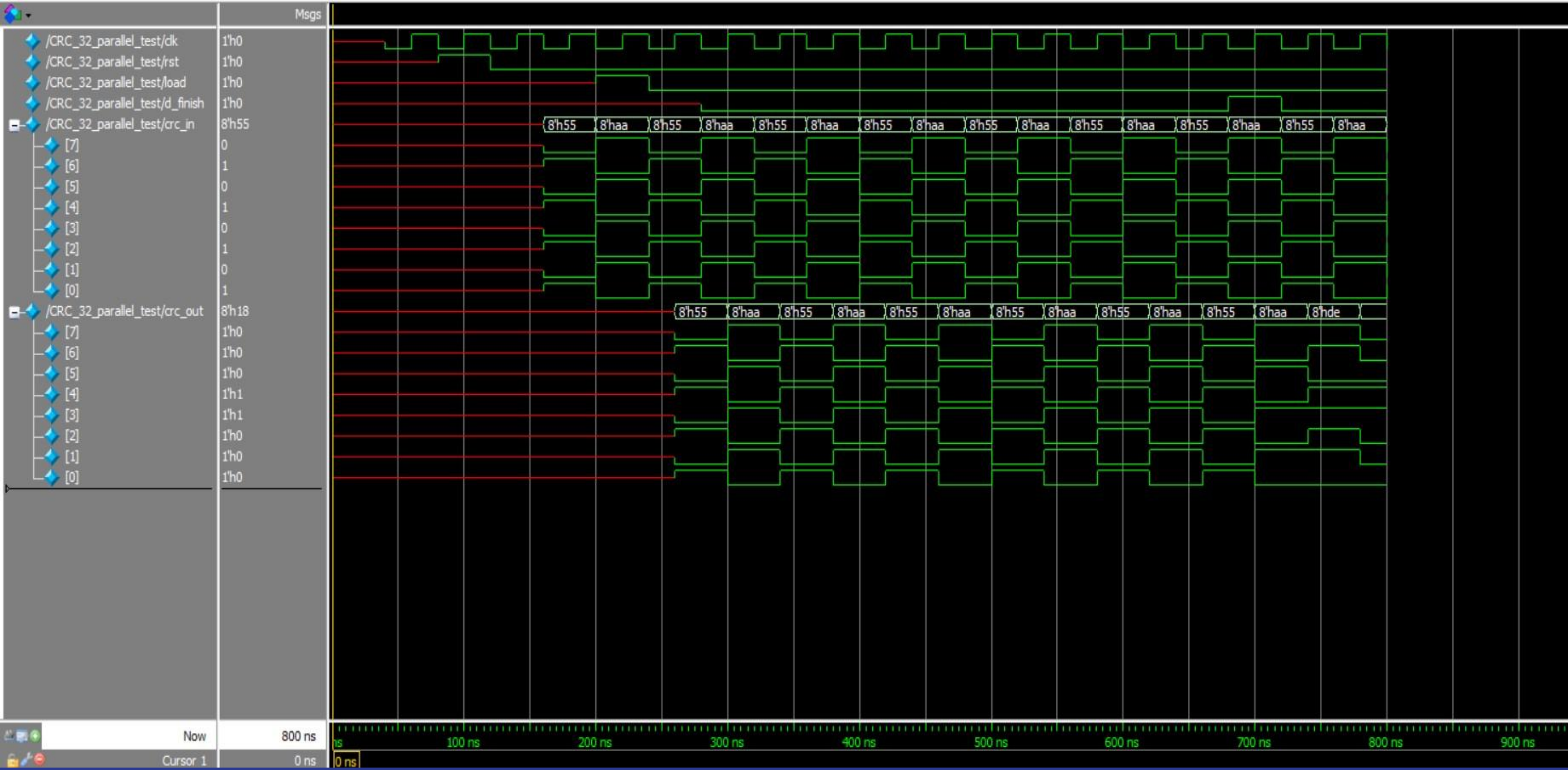
Results



```
                                  Msgs
/CRC_16_parallel_test/clk         1'h1
/CRC_16_parallel_test/rst         1'h0
/CRC_16_parallel_test/load        1'h0
/CRC_16_parallel_test/d_finish    1'h0
/CRC_16_parallel_test/crc_in      8'h55
    [7]                           0
    [6]                           1
    [5]                           0
    [4]                           1
    [3]                           0
    [2]                           1
    [1]                           0
    [0]                           1
/CRC_16_parallel_test/crc_out     8'h00
    [7]                           1'h0
    [6]                           1'h0
    [5]                           1'h0
    [4]                           1'h0
    [3]                           1'h0
    [2]                           1'h0
    [1]                           1'h0
    [0]                           1'h0

Now                               900 ns
Cursor 1                          0 ns
```

Results



```
                                      Msgs
/CRC_32_parallel_test/clk             1'h0
/CRC_32_parallel_test/rst             1'h0
/CRC_32_parallel_test/load            1'h0
/CRC_32_parallel_test/d_finish        1'h0
/CRC_32_parallel_test/crc_in          8'h55
    [7]                               0
    [6]                               1
    [5]                               0
    [4]                               1
    [3]                               0
    [2]                               1
    [1]                               0
    [0]                               1
/CRC_32_parallel_test/crc_out         8'h18
    [7]                               1'h0
    [6]                               1'h0
    [5]                               1'h0
    [4]                               1'h1
    [3]                               1'h1
    [2]                               1'h0
    [1]                               1'h0
    [0]                               1'h0

Now                                   800 ns
Cursor 1                              0 ns
```

Applications

- CRC-1: Most of the hardware, parity bit
- CRC-3: GSM mobile networks
- CRC-4-ITU:ITU-T G.704
- CRC-5-EPC: Gen 2 RFID
- CRC-5-ITU:ITU-T G.704
- CRC-5-EPC:USB Token Packets
- CRC-6-CDMA2000-A:Mobile Networks
- CRC-6-CDMA2000-B:Mobile Networks
- CRC-6-DARC Data Radio Channel
- CRC-6-GSM mobile networks
- CRC-6-ITU ITU-T G.704
- CRC-7 telecom systems, ITU-T G.707, ITU-T G.832, MMC, SD
- CRC-7-MVB Train Communication Network, IEC 60870-5

Applications

- CRC-8 DVB-S2
- CRC-8-AUTOSAR automotive integration, OpenSafety
- CRC-8-Bluetooth wireless connectivity
- CRC-8-CCITT ITU-T I.432.1,ATM HEC, ISDN HEC and cell delineation, SMBus PEC
- CRC-8-Dallas/Maxim 1-Wire bus
- CRC-8-DARC Data Radio Channel
- CRC-8-GSM-B mobile networks
- CRC-8-SAE J1850 AES3; OBD
- CRC-8-WCDMA mobile networks
- CRC-10 ATM; ITU-T I.610
- CRC-10-CDMA2000 mobile networks
- CRC-10-GSM mobile networks

Applications

- CRC-11 FlexRay
- CRC-12 telecom systems
- CRC-12-CDMA2000 mobile networks
- CRC-12-GSM mobile networks
- CRC-13-BBC Time signal, Radio teleswitch
- CRC-14-DARC Data Radio Channel
- CRC-14-GSM mobile networks
- CRC-16-Chakravarty Optimal for payloads ≤64 bits
- CRC-16-ARINC ACARS applications
- CRC-16-CCITT X.25, V.41, HDLC FCS, XMODEM, Bluetooth, PACTOR, SD, DigRF
- CRC-16-CDMA2000 mobile networks
- CRC-16-DECT cordless telephones
- CRC-16-T10-DIF SCSI DIF

Applications

- CRC-16-DNP DNP, IEC 870, M-Bus
- CRC-16-IBM Bisync, Modbus, USB, ANSI X3.28, SIA DC-07
- CRC-16-OpenSafety-A safety fieldbus
- CRC-16-OpenSafety-B safety fieldbus
- CRC-16-Profibus fieldbus networks
- Fletcher-16 Used in Adler-32 A & B Checksums
- CRC-17-CAN CAN FD
- CRC-21-CAN CAN FD
- CRC-24 FlexRay
- CRC-24-Radix-64 OpenPGP, RTCM104v3
- CRC-24-WCDMA Used in OS-9 RTOS
- CRC-30 CDMA
- CRC-32 ISO 3309 (HDLC), ANSI X3.66 (ADCCP), FIPS PUB 71, FED-STD-1003, ITU-T V.42

Applications

- CRC-32C (Castagnoli) iSCSI, SCTP, G.hn payload, SSE4.2, Btrfs, ext4, Ceph
- CRC-32K (Koopman {1,3,28}) Excellent at Ethernet frame length, poor performance with long files
- CRC-32K2 (Koopman {1,1,30}) Excellent at Ethernet frame length, poor performance with long files
- CRC-32Q aviation; AIXM
- Adler-32 Often confused to be a CRC, but actually a checksum
- CRC-40-GSM GSM control channel
- CRC-64-ECMA ECMA-182
- CRC-64-ISO ISO 3309 (HDLC), Swiss-Prot/TrEMBL,weak hashing

Advantages

- Cyclic Codes have very good performance in detecting single bit errors, double errors,an odd number of errors.

- They can easily be implemented in hardware and software.

- They are especially fast when implemented in hardware.

- This has made cyclic codes a good candidate for many networks

Thank You